Toolbox 32

For PC-1, CP-12/11/21 and LP-1/2/3 RTUs

KINGFISHER

Document Information

Document Control	
Copyright	Copyright © Semaphore (Australia) Pty Ltd. ABN 35 006 805 910 http://helpdesk.servelec-semaphore.com/, mailto:support@servelec-semaphore.com
Intellectual Property	Semaphore asserts ownership of the intellectual property contained herein and claims copyright and authorship. Semaphore has and retains all rights of ownership and use of the material herein in its on-going business.
Licence	This document is provided to the intended recipient(s) under a non-exclusive licence. This licence permits Fair Use of the document for operational requirements, without payment of further royalty or licence fee. Fair Use includes making copies of the document for operational, backup and archive purposes. Fair Use includes distributing copies of the document to other entities for the purposes of their performing related works for the intended recipient(s). Fair Use does not include creating, selling or distributing copies of the document for other purposes. All copies must retain this statement of Intellectual Property and Copyright.

Revision History		
Rev.	**Date**	**Summary**
1.23a	22/01/1996	First Release (revision number matches Toolbox software version)
1.31b	09/04/1998	Release 2
1.35b	20/07/2000	Release 3
1.41c	18/10/2001	Release 4
1.41j	17/10/2002	Release 5
1.41j-2	31/10/2002	Release 6
1.43d	30/07/2004	Release 7
1.44a	06/01/2005	Release 8
1.44d	17/11/2005	Release 9
1.45a	10/8/2006	Release 10
1.45f	31/8/2007	Release 11
1.46b	19/3/2009	Release 12.
1.46c	17/12/2009	Release 13.
1.46c-2	23/12/2010	Release 14. Added MAC address information for Ethernet ports. Added note about long file names. Added note when searching for registers in ladder logic. Added topic Redundant PCs. Added note about configuring Trio radios using TVIEW+ software. Reformatted the information in the appendix RTU Data, IO Modules. Updated descriptions for #YPMODnn, #YPPCOLnn and #YSTAT.
1.47c	28/2/2013	Release 15. Minor corrections and conversion to PDF format.
1.47d	9/8/2013	Release 16. Clearer Description for Paging operation. Hart change from Local to Network registers.
2.0	07/04/2014	Release 17. Fixed Headings and links. Added MC-12 to CP-12 conversion

Contents

Toolbox 32 1

1. Quick Start 4
2. Introduction 8
3. Getting Started 9
4. Configuration 12
5. Ladder Logic 53
6. Ladder Logic Examples 102
7. Communication Drivers 141
8. View Menu 170
9. Utilities Menu 182
10. Appendices 197

LP-2/3 Low Power RTU 1

1. Introduction 2
2. Quick Start 3
3. Ladder Logic Examples 6
4. Port Configuration 9
5. Backup Battery Calculations 10
6. LP-2/3 Registers 12
7. Specifications 16
8. Wiring Details 24
9. Appendices 26

1. Quick Start

This chapter explains how to create and download a basic configuration for a PC-1, CP-10/11/12/21 or LP-1/2/3 RTU using Toolbox 32. CP-30 / G30 RTUs are configured using Toolbox PLUS+ and ISaGRAF.

This may be a large manual but you only have to read what you need! The steps below will guide you.

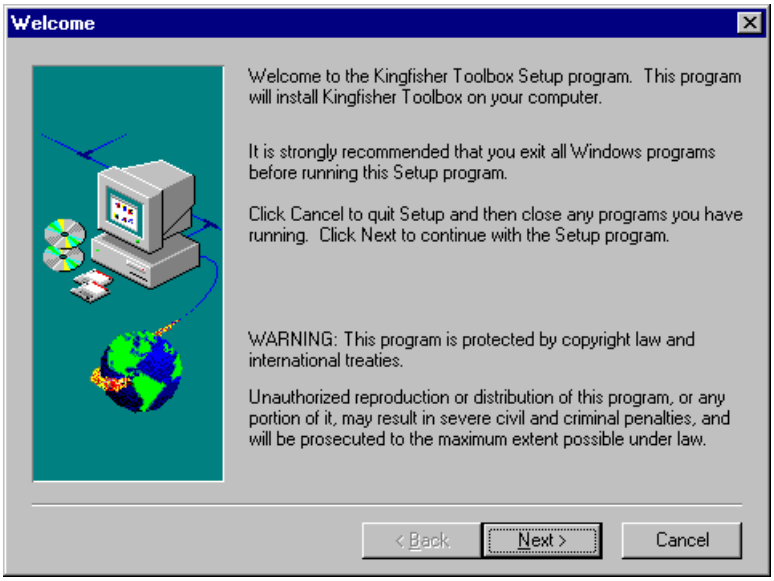
1. To Install Toolbox 32 run **SETUP.EXE**

(Toolbox 32 cannot be installed on a server or run over a network).

Latest Versions: Toolbox *upgrade* files are available from http://helpdesk.servelec-semaphore.com and can be used to upgrade a full install of Toolbox.

Toolbox menu commands and buttons to select are shown in **bold**.

Note: Windows Vista and Windows 7 do not support the Toolbox 32 Help file. To rectify please see Microsoft Support Article ID 917607 http://support.microsoft.com/kb/917607

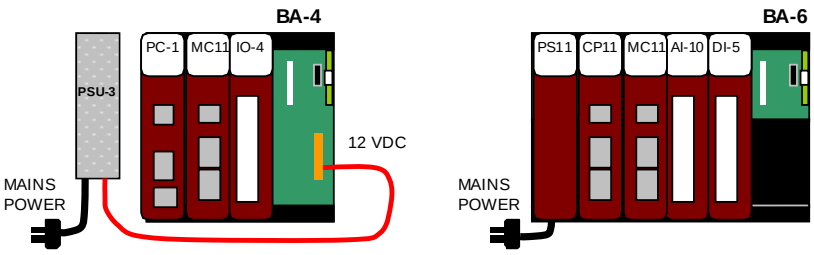

Welcome
Welcome to the Kingfisher Toolbox Setup program. This program will install Kingfisher Toolbox on your computer.
It is strongly recommended that you exit all Windows programs before running this Setup program.
Click Cancel to quit Setup and then close any programs you have running. Click Next to continue with the Setup program.
WARNING: This program is protected by copyright law and international treaties.
Unauthorized reproduction or distribution of this program, or any portion of it, may result in severe civil and criminal penalties, and will be prosecuted to the maximum extent possible under law.
< Back
Next >
Cancel

2. For LP-1/2/3 RTUs, additional configuration instructions are contained in the **LP-1 or LP-2/3 Product Manual** available from http://helpdesk.servelec-semaphore.com.

3. Power up the RTU.

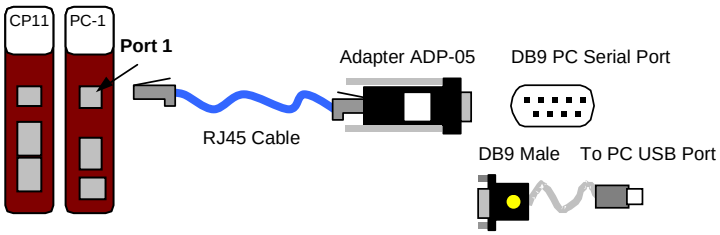
Wiring details are contained in the power supply section of the **Hardware Manual** available from http://helpdesk.servelec-semaphore.com.

For more information about assembling an RTU, please see Getting Started, Building an RTU.



4. Connect the Toolbox cable to port 1 of the RTU as shown.

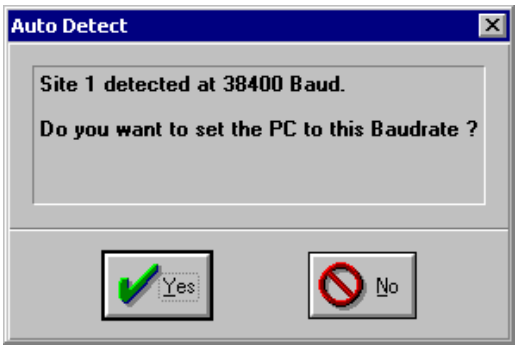
Note: if the PC only has USB ports (and no DB9 serial COM ports), a USB to 9-pin [DB9 Male] RS-232 serial adapter cable will be required.

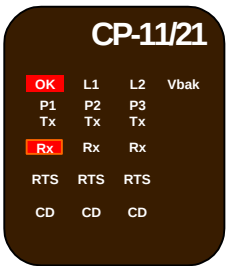


5. Start Toolbox. Ensure Toolbox is using the correct COM port in Configuration - PC Setup.

The address and baudrate of the RTU can then be automatically detected using View, Auto Detect.

For more information, please see Getting Started, Communicating With An RTU.

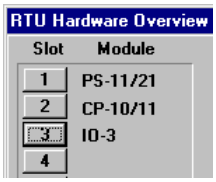


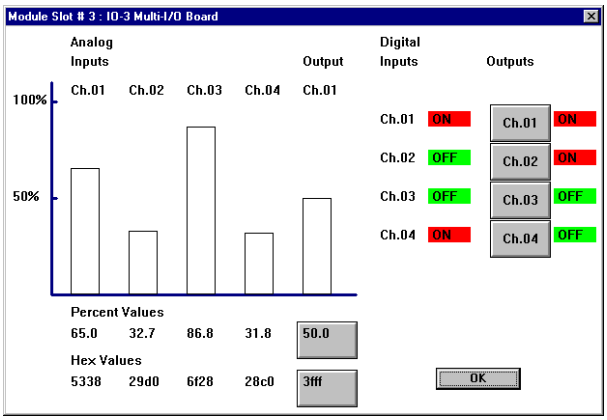


The RTU port **Rx** light will flash each time Toolbox communicates

6. If the RTU has any input/output (IO) modules, the state of all the inputs and outputs can be viewed from View, Hardware Overview.

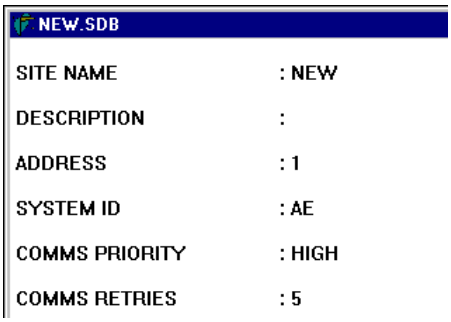
Select the **Slot** button next to the module to view details.





7. A new RTU configuration file is created by selecting **File, New**.

The new configuration file can be saved using an appropriate name by selecting **File, Save As**.



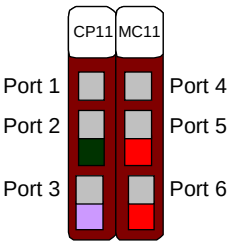
8. The following items are all configured from the Configuration menu.

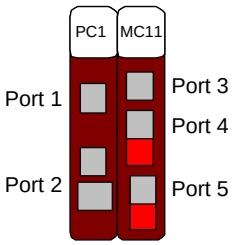
The default settings that need to be changed are detailed.

Address and Description: Set Site Address to a number in the range of 1 to 249 eg. 1.

System Parameters: Set RTU/CPU Type to match the processor that is being used eg. CP-10/11.

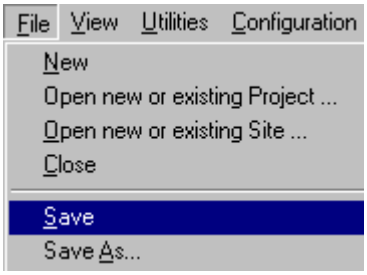
Port List: Select button **2** if there is a port 2 option board installed. The default port settings are for an RS232 serial option port (labelled SER-I or SER-S). Select **OK** if using RS232 or change **Type** to match the type of option board. For more information on the various port settings please see the topic Configuration, Port List. Note: if there are more communication ports, these will also need to be configured. Usually each port is assigned the next available port number as illustrated below.





Network List: Only used if communicating with other RTUs. Please see the topic Configuration, Network List for more information.

9. Save the configuration by selecting **File, Save**.



10. Extra functionality is added to the RTU using firmware drivers.

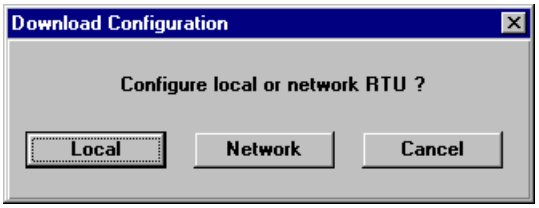
Drivers include Modbus, DNP3, SMS paging, AGA calculations and many more. A driver listing (**protocols.pdf**) is available from http://helpdesk.servelec-semaphore.com.

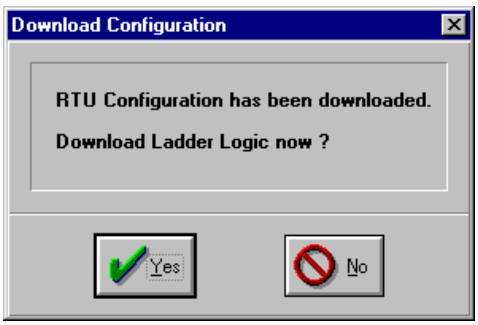
Standard drivers are available from the website and can be downloaded by selecting **Utilities, Advanced, Download Firmware Driver**.

11. Download the configuration file (Filename.SDB) into the local RTU using **Configuration, Download RTU Configuration**.

If asked "Configure local or network RTU?" select **Local**

When asked "Download Ladder Logic now?" select **No** (there isn't any ladder logic yet!)



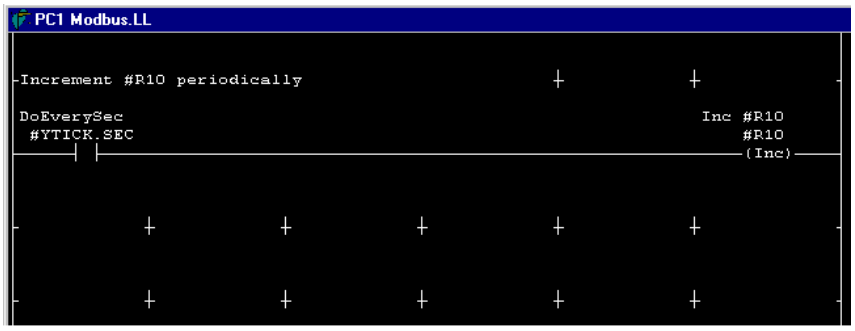


12. Ladder logic is used to add more intelligence to the RTU.

Basic RTU configurations that have ladder logic are available from http://helpdesk.servelec-semaphore.com.

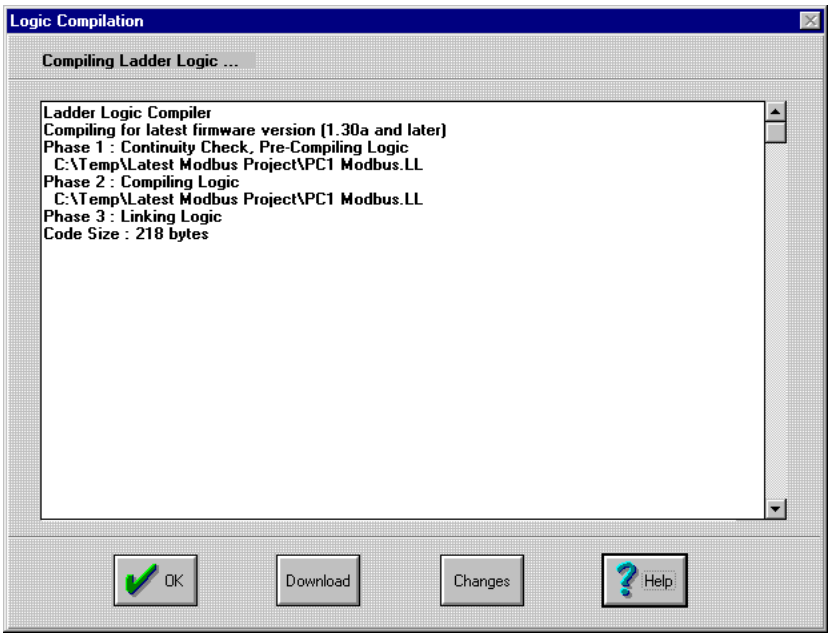
Explanations about ladder logic and how to create it are contained in the Ladder Logic chapter.

Sample ladder logic code is contained in the Ladder Logic Examples chapter.



13. Ladder logic must be compiled before it is downloaded.

First select **Ladder** (or **Logic**), **Compile** and then select **Ladder** (or **Logic**), **Download To RTU**.

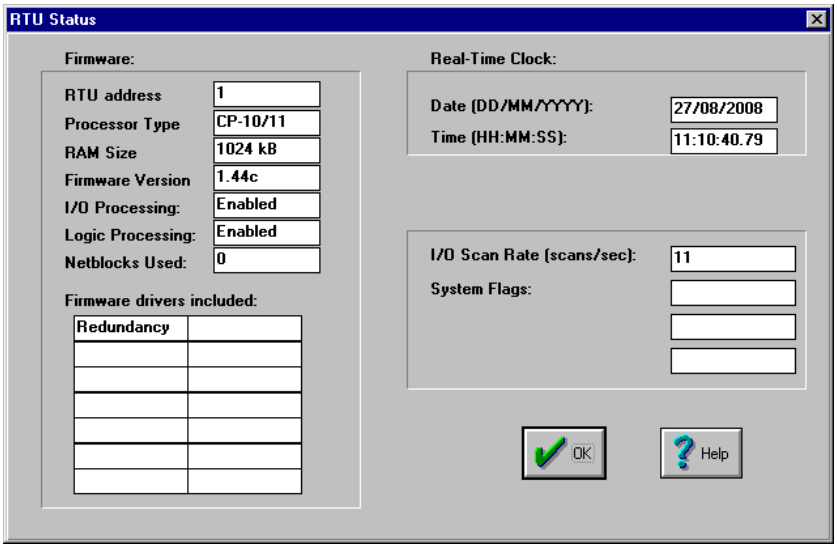


14. The RTU is now configured and ready for operation!

The RTU can be checked by selecting **View, RTU Status**.

Check that **I/O Processing** and **Logic Processing** are both enabled and that the RTU time is updating and correct.

The clock can be set from **Utilities, Set Real Time Clock**.



2. Introduction

An RTU is electronic equipment that contains a computer. RTUs are often located in remote places. This led to the name *Remote Telemetry Unit* (RTU). RTUs can be wired to a whole range of devices like switches, relays and sensors. Each RTU can monitor and control the things it is wired to.

There are two types of devices that an RTU can be wired to - digital and analog. A digital signal is an ON or OFF state of a switch or a relay. An analog signal is a variable measurement like tank level or temperature.

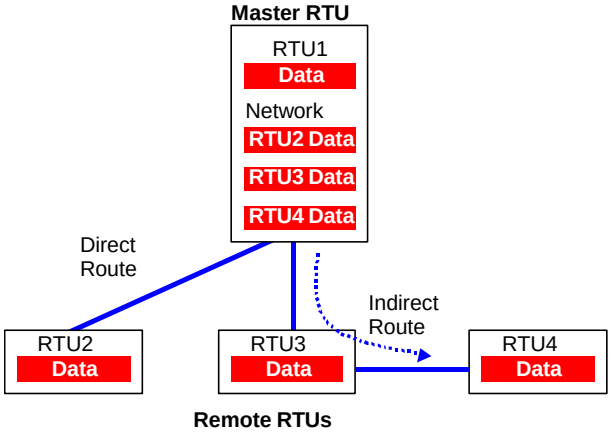
An RTU can also obtain data by communicating with intelligent devices (eg. a PLC).

RTUs can be *programmed* to carry out a wide range of tasks. For example:

- Set outputs according to the state of inputs (eg. run a pump to fill a tank that is low)
- Send a message when there is new data or a significant event has occurred
- Perform complex mathematical calculations

Kingfisher RTUs have continued to increase in *speed and power* over the years and can now provide:

- Data Logging
- Alarming - auto dialing, SMS messages
- Diverse communications - data radios, dialup and cellular modems, leased line, Ethernet and more…
- PLC-like logic processing
- Massive networks - more than 65,000 RTUs
- Support for various protocols eg. Modbus, DNP3



An *RTU network* is 2 or more RTUs that can communicate with each other in some way. The communication path is called a *route*.

Usually one RTU is setup as the *Master RTU*. The master RTU regularly polls data from all the other RTUs. The other RTUs are referred to as *Remote RTUs* and can report data changes as they occur (called exception reports). RTU configuration is completely flexible and allows for many other types of communication setups.

This example shows RTU1 as the master and RTUs 2-4 as remote RTUs. RTU3 also stores and then forwards messages between RTU1 and RTU4.

If only *polling* is used, it will take up to the regular polling interval before the master RTU knows about new data from remote RTUs. Eg. If the master polls the remote RTUs every 2 minutes, it will take up to 2 minutes before the master receives new data from the remote RTUs. If only *exception reporting* is used, the master RTU will not know if a remote RTU has failed. If the master RTU does not receive a message from a remote RTU for a long time this could mean that either there is no new data or that the remote RTU has stopped communicating.

The best method is to use both polling and exception reports. This means that if a remote RTU fails, the master RTU will find out about the fail when it performs the next poll. And as soon as data changes or a significant event occurs, the master RTU will be notified by an exception report.

Kingfisher PLUS+ RTUs (not CP-30 / G30) have three places for storing data: Hardware Registers (hardware inputs and outputs), Local Registers and Network Registers. When data is sent from one RTU to another, the data is always stored as network registers in the destination RTU. Network registers are simply a copy of the hardware and local registers that are received from the sending RTU.

Kingfisher PLUS+ RTUs (except LP-1/2/3 and G30) are comprised of up to 64 modules that plug onto 1 or more backplanes.

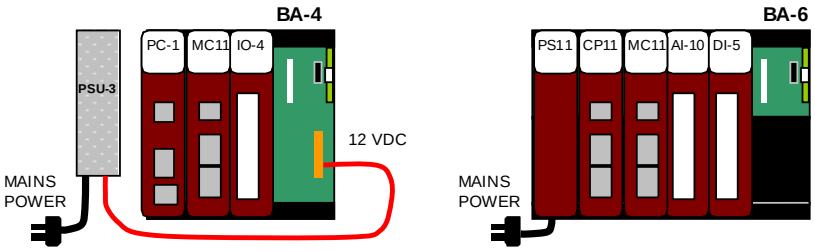
3. Getting Started

Building An RTU

Every RTU must have a power supply and a processor (CPU) module. Most RTUs also have some inputs and outputs as well. The two main types of power supplies are a PSU-3 and a PS-11/21. The two main types of processors are a PC-1 and a CP-11/12/21. A PSU-3 is used with a PC-1 and a PS-11/21 is used with a CP-11/12/21 as shown below. A wide variety of input and output modules are available and these can be used in any combination.

Both types of RTUs are configured the same way with the same Toolbox software. However, the PC-1 RTU was designed to be a simpler RTU and does not have the same speed, memory and functionality of a CP-11/12/21 RTU.

A PC-1 module can only be used on a 4-slot backplane (a BA-4) and is installed in the left-most slot. A CP-11/12/21 can be used on a 4, 6 or 12 slot backplane (a BA-40, BA-6 or BA-12 respectively) and can be placed in any slot. Usually the CP-11/12/21 is installed in slot 2 (the second slot from the left) and the PS-11/21 power supply is installed in slot 1. Typical RTU layouts are shown below.



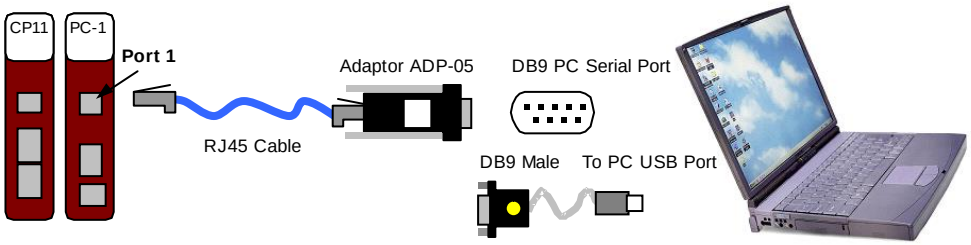
A number of backplanes can be linked together to allow one processor module to control up to 63 IO modules.

⚠ **Caution!** IO Modules can be installed or removed while an RTU is running (called Hot Swapping) without any effect on the other modules. However, adding or removing an MC-xx module will cause a warm start to occur which may temporarily affect the state of any outputs. For more information about hot swapping power supply or processor modules please see the Redundancy appendix.

Communicating With An RTU

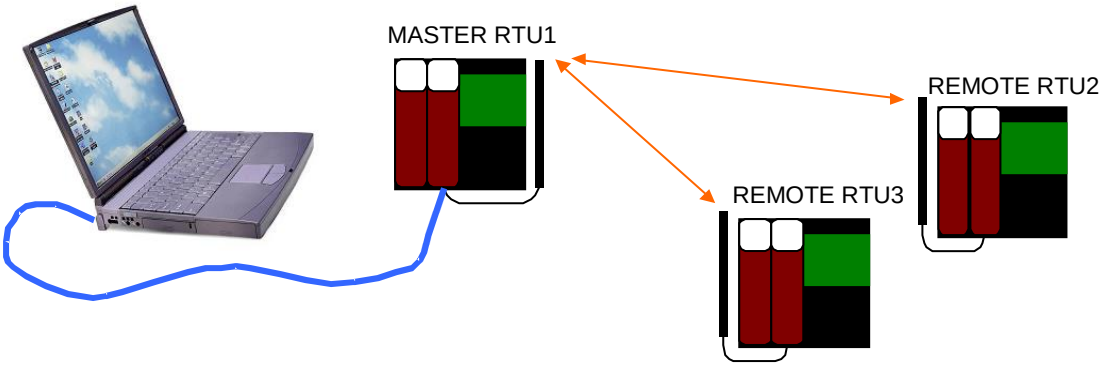
Local RTU

- Connect the Toolbox cable to port 1 of the RTU as shown below. Note: if there are only USB ports on the PC (and no DB9 serial COM ports), a USB to 9-pin [DB9 male] RS-232 serial converter cable will be required.

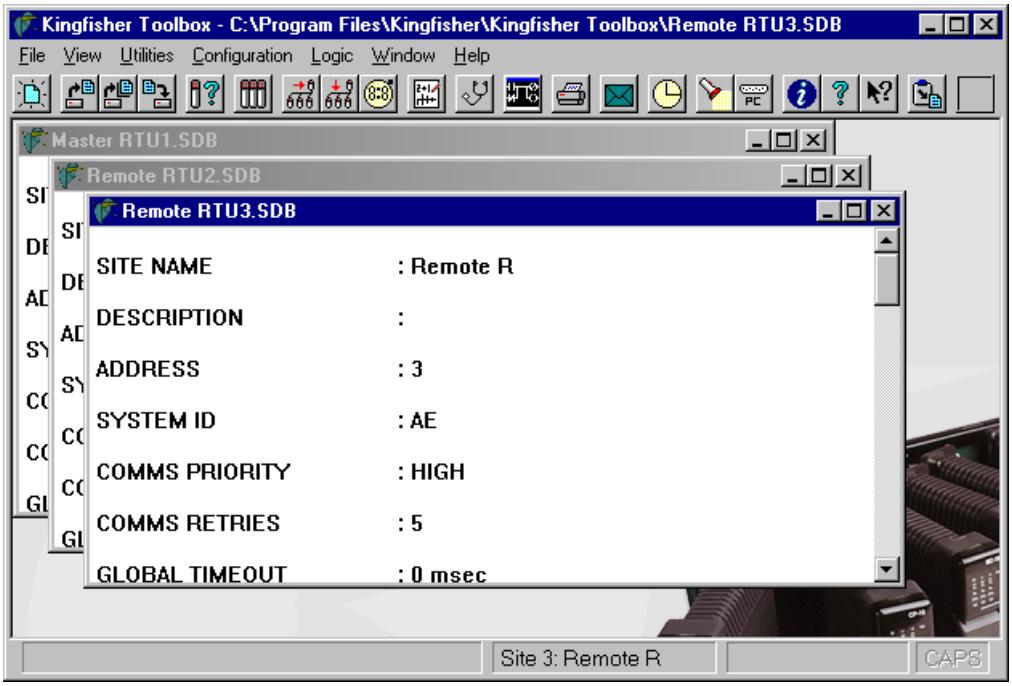


- COM port 1 and a baud rate of 9600 are commonly used between the PC and RTU. If COM1 is being used by another device, COM2 to COM48 can be used to communicate with the RTU instead. To change the port used by Toolbox, select Configuration, PC Setup.
- With no Site configuration loaded in Toolbox, select View, Auto Detect. Toolbox will then attempt to detect the address of the local RTU. If the COM port is working OK, the RX light of the RTU port will flash each time Toolbox attempts to communicate.
- Auto Detect will first try to communicate at the baud rate configured in PC Setup. If the baud rate is *correct*, Auto Detect will return the address of the RTU. If the baud rate is *incorrect*, Toolbox reports “No RTU detected at current baud rate. Continue search at other baud rates?” Select **Yes**. Toolbox will then try all the baud rates to communicate with the RTU and when successful will return the address and baud rate of the local RTU port. When asked “Do you want to set the PC to this baud rate?” select **Yes**.
- The local RTU will only respond to messages that target its own address or address 0. All RTUs respond to address 0. Toolbox uses address 0 when no Site configuration is open. If a configuration is opened or created, Toolbox will then use the address of the selected configuration when communicating with the RTU.
- To check the status of the RTU, select **View, RTU Status**. The time field will keep updating every time Toolbox polls the RTU. The Comms Repeat Rate setting in PC Setup determines the update rate.
- The modules that are installed on the backplane are automatically detected by the RTU and can be viewed using **View, Hardware Overview**. By selecting the button next to any of the modules that appear, the module input and output (IO) details are displayed on the screen. Outputs can then be set using Toolbox if they are not being controlled by ladder logic.

Remote RTU



- Once communications with the local RTU have been established (please see above), it is possible to communicate with remote RTUs that are connected to the local RTU. This will only work if the local RTU has been configured with the appropriate network links (please see the topic Configuration - Network for more information).
- Load the RTU configurations for the remote RTUs. If the configurations are not available, basic configurations can be created in Toolbox (**File, New**) for each remote RTU. Only the address of each new configuration needs to be updated to correspond to the address of each remote RTU (**Configuration, Address & Description**). Configurations that correspond to the above network are shown below.



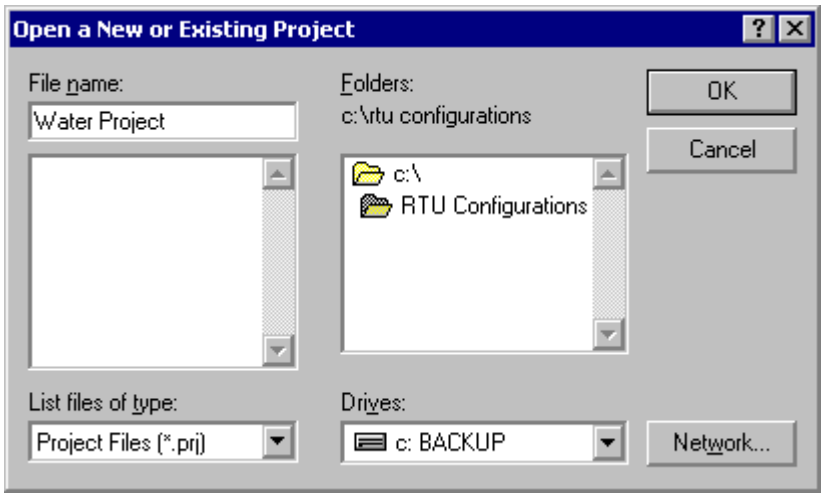
- Toolbox determines which RTU to communicate with according to which configuration window is active (highlighted or in front). As shown above, the configuration for Remote RTU3 is active. Toolbox will then attempt to communicate with Remote RTU3 when commanded (eg. select **View, RTU Status**). All the commands that can be sent to a local RTU can also be sent to a remote RTU.
- Comms fail: it takes longer to receive a reply to a message sent to a remote RTU. Toolbox may timeout before the reply is received and flag a comms fail. To prevent comms fails, select **Configuration, PC Setup** and set **Comms Timeout** to 5 or more seconds and set the **Number Of Retries** to 1 or greater.

4. Configuration

New Project	Optional: group RTU configurations in a project
New Site	Create an RTU configuration file
Address & Description	RTU address and site description
System Parameters	General RTU settings
Memory	Define memory usage
Port List	Add ports and configure port settings
Network List	Optional: how the local RTU communicates with remote RTUs
I/O Modules List	Optional: custom I/O module settings
Phone Directory	Optional: only used for PSTN or GSM modems
TMR Directory	Optional: only used for trunk mobile radios
Pager Configuration	Optional: SMS or radio paging
PC Setup	Toolbox communication settings
Download	Download RTU site configuration file (ladder logic is a separate file)

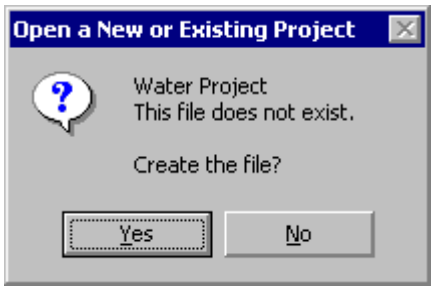
Configuration - New Project

A project is used to group all the RTU Configurations in a telemetry system. To create a new project select **File, Open new or existing Project...** The window below will then be displayed.

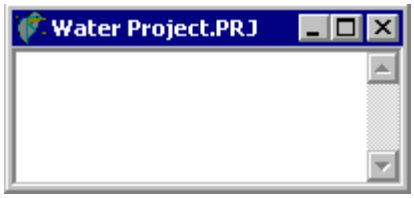


First double-click on the yellow Folders to select where to store the new project. The project path can have up to 240 characters and may contain spaces. Then enter the file name and select **OK**. The file name can have up to 130 characters and may contain spaces. It cannot contain any of the following characters **\ / : * ? " < > |**

The window below will then be displayed:



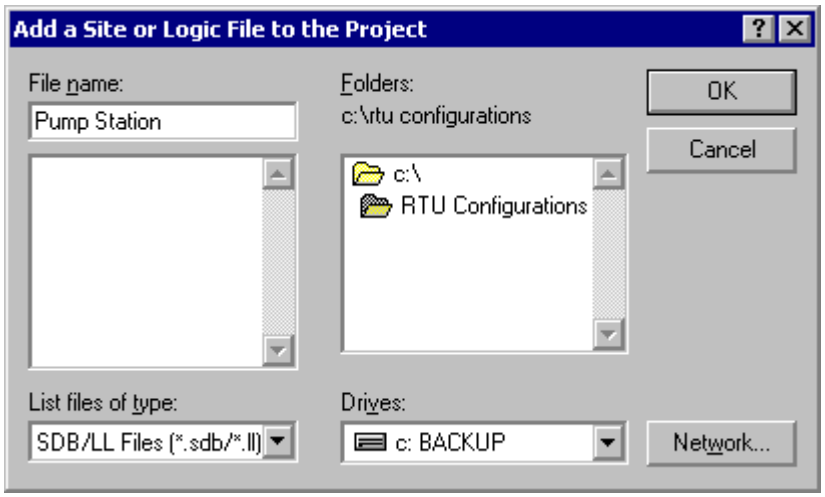
After selecting Yes, the window below will be displayed.



Finally, select **File, Save** to create the project file (Filename.PRJ).

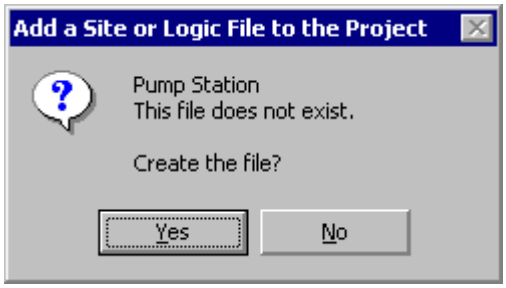
Configuration - New Site

After a project has been created or opened, an RTU site can now be created. An RTU site (Filename.SDB) is a text file that contains all the communication settings for the RTU. To create a new site, select **File (or Project), Add a Site or Logic File**. The window shown below will then be displayed.

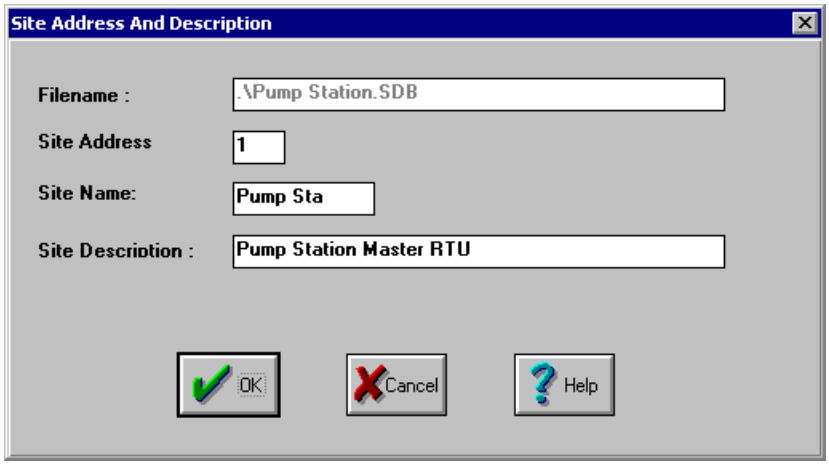


Enter a file name with no extension eg. Pump Station (as shown above) and then select **OK**. The file name can have up to 130 characters and may contain spaces. It cannot contain any of the following characters **\ / : * ? " < > |**

The window below will then be displayed:

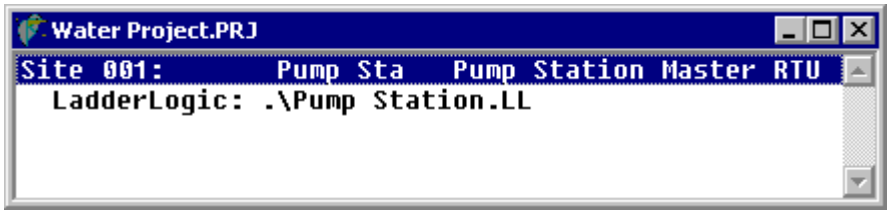


After selecting Yes, some more information will be requested. An example window with the information filled in is shown below:



The above information should be filled in according to the next topic - Configuration - Address and Description.

After selecting OK, the new site will now appear in the project window as shown below.



When a Project file is open (ie. it is the active window), RTU sites can be added or removed from the project by using the menu **Project, Add A Site Or Logic File or Project, Remove Site**.

By double-clicking on "**Site 001: …"** etc in the project window, the **Pump Station.SDB** file will appear. The RTU settings can then be configured using the **Configuration** menu.

An RTU is configured by stepping through each of the **Configuration** options and filling in the appropriate information as detailed in the following sections.

Configuration - Address & Description

Each RTU in the telemetry system should have a *unique* address. This allows the RTUs to communicate with each other one at a time and avoids communication fails.

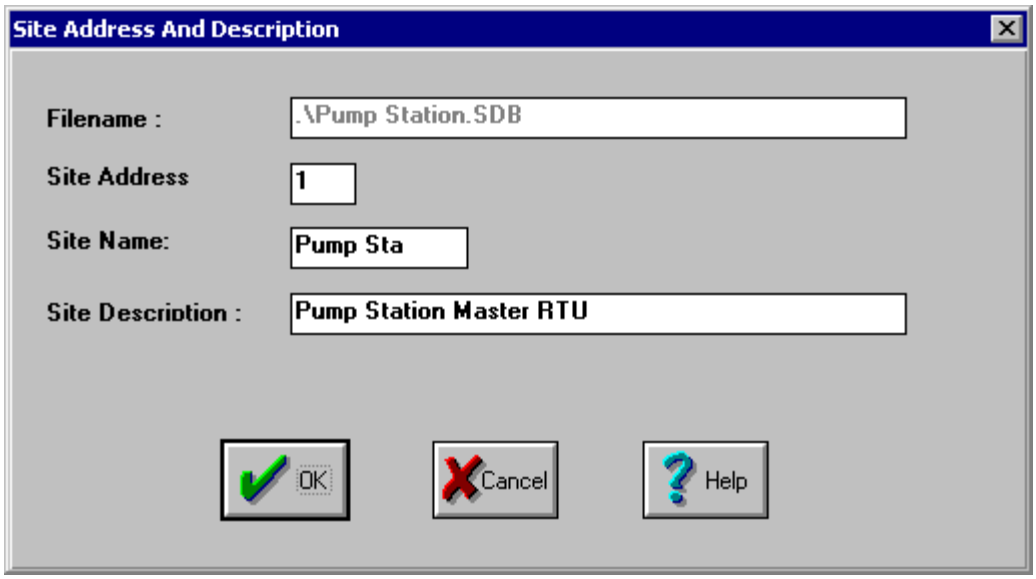


Figure: Address And Description window (displayed if a project is being used)

Site Address: (0-249) Address 250 is reserved for paging parameters and addresses 251 to 255 are reserved for Toolbox (Configuration, PC Setup, PC's Network Address) as a PC running Toolbox is treated like an RTU. Address 0 should not be used for an RTU in a network as address 0 is the global address. Every RTU (no matter what its address is) will respond to messages for RTU 0. Address 1 is commonly used for the master RTU.

Site Name: (Only visible when a project is open) 8-character site description. By default, the first 8 characters of *Filename.SDB* are used.

Site Description: (Optional) A 32-character comment used to describe the RTU site.

Configuration - System Parameters

Use to configure general RTU settings.

Please ensure **RTU/CPU Type** is set to the type of processor module that is being used. The default values for the other parameters can be used in most cases and are detailed below.

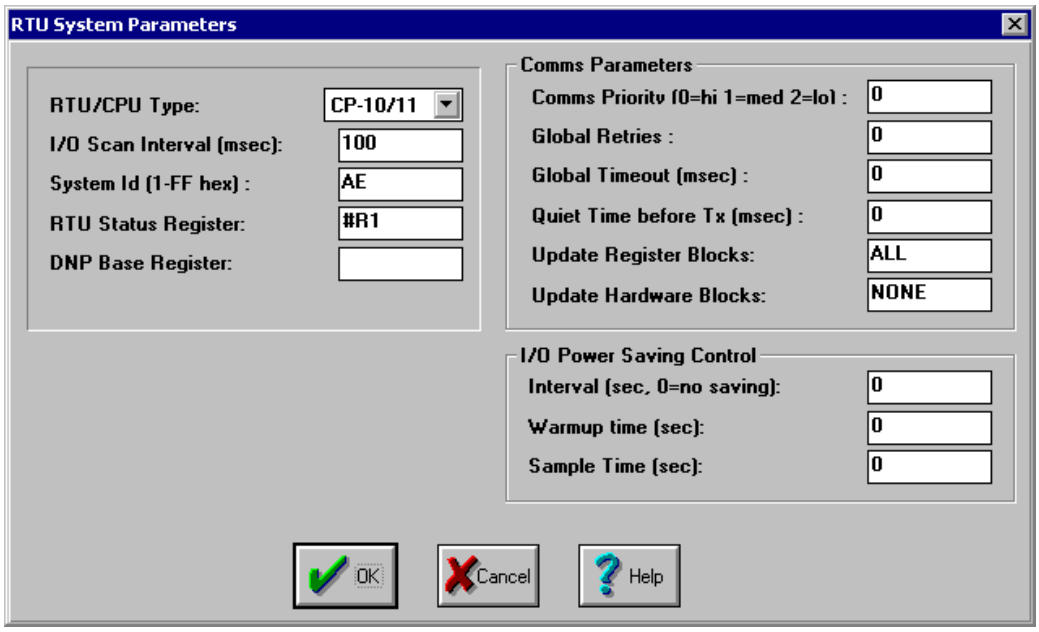


Figure: Example System Parameters For A Master RTU

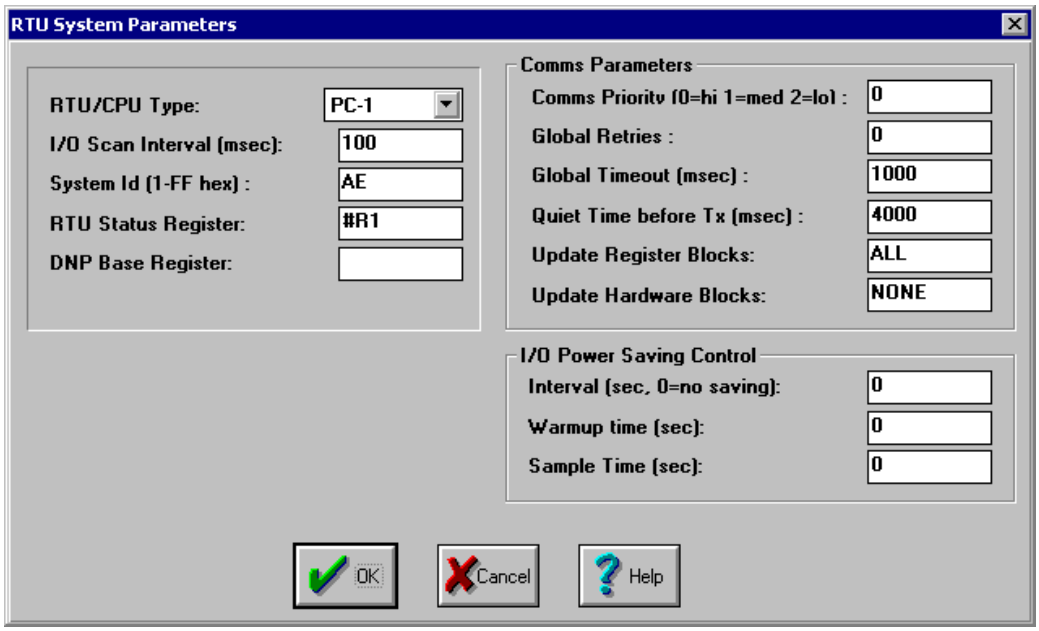


Figure: Example System Parameters For An Outstation RTU

RTU Type: The type of processor used by the RTU: *CP-1*, *PC-1*, *CP-10/11*, *CP-21*, *CP-12, SBX*, *ERS Micro*, *LP-1/2/3*, *SB-1*, *Micro-4* or *Other*. This setting allows Toolbox to compile ladder logic in the correct format (for each microprocessor type). Note a CP-30 or G30 is configured using Toolbox PLUS+ software.

I/O Scan Interval: The period (in milliseconds) at which the RTU scans all its IO modules and processes ladder logic (default = 100 ms). By increasing the I/O Scan Interval, more time is left over for the RTU to respond to incoming messages on communication ports. Set I/O Scan Interval to 10 ms or less if wanting to use Timer blocks and count in 100ths of a second (provided that the RTU is capable of scanning that fast). Note: a zero I/O Scan Interval will cause the RTU to scan as fast as possible and cause a *Scan Overrun* system flag to appear in the RTU Status. Scan Overrun means that the RTU cannot process all the IO and ladder at the configured interval.

System ID: (Default = AE Hex) The communications sync character used to screen incoming Kingfisher messages. An RTU will only respond to Kingfisher messages that have the same sync character as this System ID or that begin with AE (Hexadecimal). It is recommended that a system ID of *AE* is used except when configuring a store-and-forward RTU as detailed in the topic Configuration - Network List, System ID. System ID is not used by other protocols.

RTU Status Register: Blank or a local register (#R). If a local register is entered, the RTU system status register (#YSTAT) will be continuously copied into this register even if ladder logic is disabled. This option makes it possible to monitor the RTU status and alarm if the IO Processing and/or Logic Processing is disabled or corrupted.

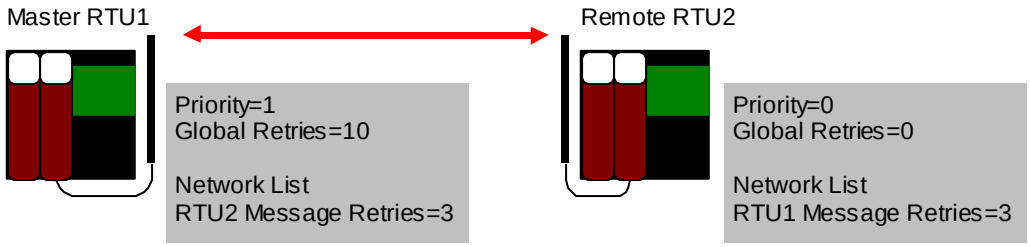
DNP Base Register: (#R) A local register defining the beginning of a block of 64 registers (Semaphore uses #R129 by default). All the DNP3 parameters are defined in these registers by using ladder logic. The definition of each register in the 64-register block is available from Semaphore by request.

Comms Priority: (0, 1 or 2) Comms Priority defines how often the RTU tries sending messages after a communications failure. A Comms Priority of 0 is used by default. The **Message Retries** parameter mentioned below is configured for each RTU in Configuration - Network List.

- **Priority 0:** Each message will have up to Message Retries or until it is successful. The Global Retries field is ignored.
- **Priority 1:** Each Message will have up to Message Retries or the amount specified in the Global Retries field (whichever is less). Once communications to an RTU have failed more than the Global Retries setting, the next messages will only be sent once until a successful message is received from that RTU.
- **Priority 2:** Each Message will have up to Message Retries or the amount specified in the Global Retries field (whichever is less). Once communications to an RTU have failed more than the Global Retries setting, no more messages will be sent to that RTU until a successful message is received from that RTU.

Global Retries: The total number of consecutive communication attempt failures for any RTU site before action is taken. This field is only used when a Comms Priority of 1 or 2 is configured (please see above).

Eg. After 10 communication fails in a row to RTU2, RTU1 will then only have one attempt at each new message to RTU2. RTU2 will continue to have up to 3 retries at each new message to RTU1. The global retries setting for each RTU will be:



Global Timeout: This parameter is used with the Timeout setting of the Network List for each network RTU. An RTU will wait for Timeout milliseconds for a reply to its first message attempt. If more attempts are required, the RTU will wait a combination of Timeout and Global Timeout milliseconds before initiating the next attempt or recording a message failure as detailed in the Timeout section of the Network List.

Quiet Time before TX: [Applicable to radio communications only] The minimum amount of quiet time [carrier signal not detected] that an RTU will wait before sending a message. Used to implement priority in a radio system to prevent message clashes. Quiet Time Before TX is usually set to 0 in the master RTU. When used in store-and-forward RTUs, messages are delayed by this time before being forwarded and so a Quiet Time of 1000 ms is typically used in store-and-forward RTUs.

Update Register Blocks and **Update Hardware Blocks:** (ALL, NONE, 16#0000 to 16#FFFF) The local register blocks and hardware register blocks that will be checked and polled (if they have changed) when an Rx Update message is received from another RTU. A hexadecimal constant (please see the appendix - Hexadecimal Numbers) can be entered corresponding to the blocks to update. Alternatively, *ALL* (all blocks are updated) or *NONE* (none of the blocks are updated) can be entered. When *ALL* is selected (or 16#FFFF hex) for Update Register Blocks, all 2048 local registers are updated. Note: a maximum of 16 register and hardware blocks can be checked and updated for each RTU. Ie. If *ALL* is specified for Update Register Blocks, then *NONE* must be specified for Update Hardware Blocks and vice versa.

Update Register Blocks Constants			
Block	**Registers**	**Ch**	**Constant**
1	#R1 to #R64	1	0001 Hex
2	#R65 to #R128	2	0002 Hex
3	#R129 to #R192	3	0004 Hex
4	#R193 to #R256	4	0008 Hex
5	#R257 to #R320	5	0010 Hex
6	#R321 to #R384	6	0020 Hex
7	#R385 to #R448	7	0040 Hex
8	#R449 to #R512	8	0080 Hex
9	#R513 to #R576	9	0100 Hex
10	#R577 to #R640	10	0200 Hex
11	#R641 to #R704	11	0400 Hex
12	#R705 to #R768	12	0800 Hex
13	#R769 to #R832	13	1000 Hex
14	#R833 to #R896	14	2000 Hex
15	#R897 to #R960	15	4000 Hex
16	#R961 to #R1024	16	8000 Hex

Update Hardware Blocks Constants			
Hardware	**Registers**	**Ch**	**Constant**
Analog Modules 1-8	#A1.1 to #A8.8	1	0001 Hex
Analog Modules 9-16	#A9.1 to #A16.8	2	0002 Hex
Analog Modules 15-24	#A17.1 to #A24.8	3	0004 Hex
Analog Modules 25-32	#A25.1 to #A32.8	4	0008 Hex
Analog Modules 39-40	#A33.1 to #A40.8	5	0010 Hex
Analog Modules 41-48	#A41.1 to #A48.8	6	0020 Hex
Analog Modules 49-56	#A49.1 to #A56.8	7	0040 Hex
Analog Modules 57-64	#A57.1 to #A64.8	8	0080 Hex
Digital Modules 1-64	#D1.1 to #D64.16	9	0100 Hex
		10-16	N/A

I/O Power Saving Control

Configures the RTU to switch on and off various output voltages (not supported by LP-x RTUs). This enables the RTU to reduce power consumption by switching off external devices. Analog and Digital input values are frozen (maintained) while the output voltages are switched off.

The voltages that can be controlled are:

- 24 V output voltage from IO modules (eg. AI-1, IO-3, IO-4)
- 24 V auxiliary supply from a PSU-1, PS-1, PS-xx or PC-1
- 12 V Vr from a PC-1, PS-1 or PS-xx (note: the 12V output supply from an LM-2 or from a PC-1/MC-1 radio board is not switched off)

The IO does the following while the output voltages are switched off:

- Digital Inputs are frozen
- Analog Inputs are frozen
- Analog Outputs are switched off (0 mA loop current)
- Digital Outputs are unaffected and can still be switched on and off

For example, an RTU can be configured to switch its output voltages on after 5 minutes, wait two seconds, scan all the IO modules for 10 seconds and then switch off its output voltages. The time intervals used in this process are configured using the following parameters:

- **Interval:** The period of time (in seconds) that the output voltages are switched off for. To disable Power Saving Control, set Interval to zero.
- **Warmup Time:** The period of time after the output voltages are switched on that is waited before scanning the IO modules.
- **Sample Time:** The period of time that the IO modules are scanned for after the Warmup Time has expired. The Power Saving Control cycle is then repeated Interval seconds after the Sample Time has expired.

During the Interval and Warmup Time periods, the IO is not scanned. The IO values obtained from the last Sample Time period are maintained.

Configuration - Memory

Each RTU has battery backed RAM (SRAM) that is used to store information about that RTU. Some memory is reserved by the RTU for storing module data, local registers, configuration parameters and other information used by the operating system. The remaining memory is configurable and is used to store event logs, ladder logic, network data, phone numbers, images and firmware drivers. The memory available for user configuration can be checked by selecting the **Check memory usage** button or by referring to the table below.

CPU	Reserved	User	Total
PC-1* / CP-1	64K	192K	256K
CP-10/11	576K	448K	1024K
CP-21	680K	1344K	2048K
LP-1/2/3	128K	384K	512K
CP-12	64K	448K	512K

* Early revision PC-1 modules had a total of 128K SRAM.

Memory space is automatically allocated for telephone numbers (if configured) and the ladder source file (if stored in the RTU) when the RTU configuration is downloaded. The default Memory configuration window is shown below.

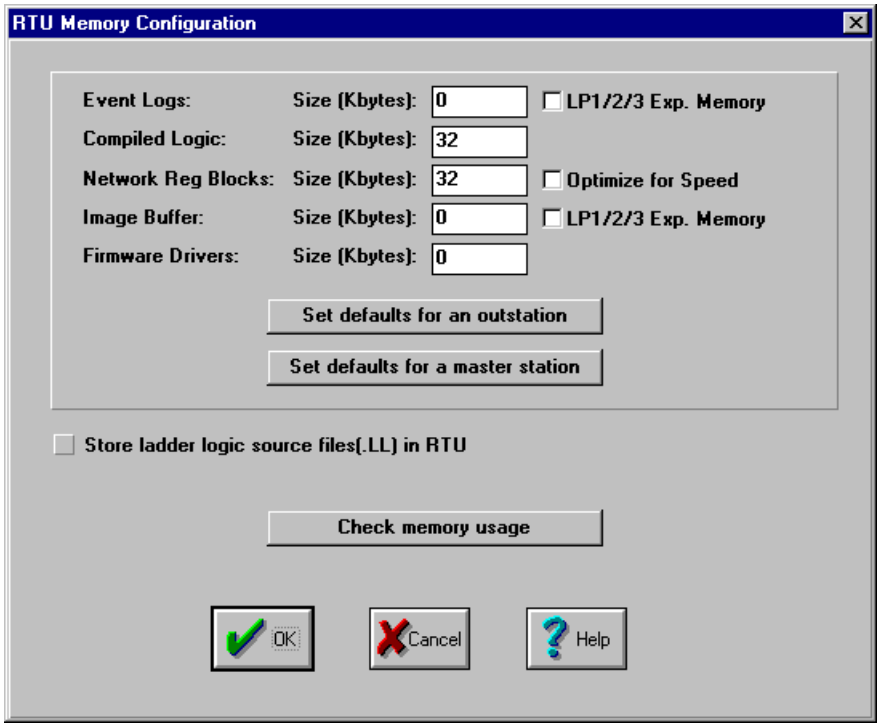


Figure: Default RTU Memory Configuration Window

Event Logs: Each event log is stored using 12 bytes. The maximum number of event logs the RTU can store is [Event Logs Size (Kb)] x 1024 / 12 bytes. Eg if 32 Kb is allocated for event logs, maximum number of event logs = 32 x 1024 / 12 = 2730. Note: a CP-21 can store a maximum of 65535 event logs. An LP-1/2/3 can store an approximate maximum of 300,000 event logs when using expanded memory.

Compiled Logic: the amount of memory to use for the compiled ladder logic file (*FILENAME.LLO*). The size of the compiled ladder logic is displayed when the logic is compiled. 32 Kb is enough memory for most applications.

Network Reg. Blocks: Set to 1 or greater to allow the RTU to store data from a remote RTU. 1K per outstation RTU is usually enough. If the network data is overflowed, a **Netblocks Overrun** error will be triggered (please see #YFLAGS.9) and displayed in the RTU Status. When the RTU is cold-started, all the network blocks are cleared.

Calculating the Network Register Blocks Required

Analog, digital and local register information is all stored in separate network register blocks. For analog data (#AI or #AO), 8 registers are used to store each module's inputs or outputs while for digital data (#DI or #DO) a single register is used for each module's inputs or outputs. Each network block contains 64 registers (128 bytes). This means that one network block can store the data for 8 consecutive analog modules or for 64 digital modules or for 64 consecutive local registers. The first network block contains registers 1 to 64, the second network block contains registers 65 to 128 and so on. This means that in order to store #R64 and #R65, two network blocks would be required as the first block is used to store #R64 and the second block is used to store #R65. Similarly, with analog registers, the first network block contains the data for analog modules 1 to 8, the second network block contains the data for analog modules 9 to 16 and so on. Example:

Data To Be Stored	Network Blocks Required
#DI1, #DI4, #DO5	1
#AI1.4, #AI3.1 to #AI3.8, #AI9.1	2
#R1, #R2, #R100	2

Optimize For Speed: (For CP-xx only) When ticked, the RTU optimizes the network register (**#**NR…) access speed by using a lookup table. This will allow the scan rate of IO modules and ladder logic to be vastly improved (up to twice as quick) if ladder logic uses a lot of network registers. Note: when using this option, ensure at least 17K of memory is allocated for Network Reg. Blocks as 16K of this memory is used for the lookup table. The remaining memory is used for storing network registers. Does not apply to Network Analog (**#**NA…) or Network Digital (**#**ND…) registers.

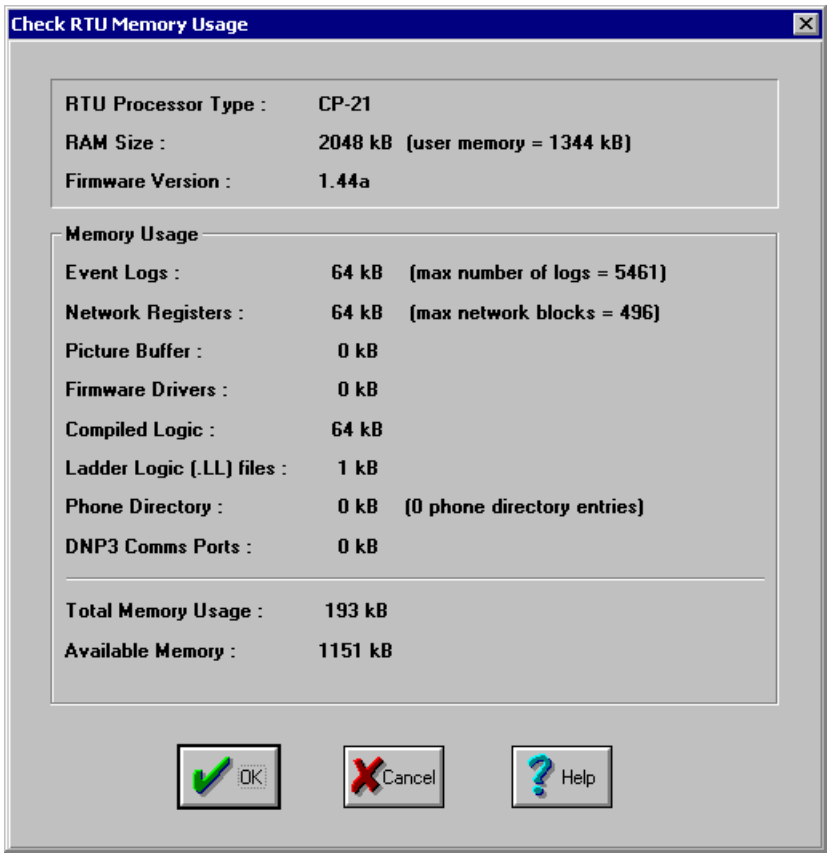
Image Buffer: (For image capture option boards only). A medium quality image uses 10 kB of memory. It is recommended that at least 100 kB be allocated for image storage to allow buffering before the images are uploaded. When the buffer is filled, the oldest image is over-written by the newest image. When using an MC-xx for image capture, 256kB of memory is automatically allocated for image storage in the MC-xx and the Image Buffer setting can be set to O kB.

Firmware Drivers: If there is not enough room in flash memory, drivers can also be downloaded to Static RAM (SRAM). Firmware drivers stored in SRAM are erased after a cold start (drivers are not affected by a cold start when stored in flash memory). LP-1/2/3 RTUs only store drivers in SRAM.

LP1/2/3 Exp. Memory: An LP-1/2/3 has 384K of SRAM available for user allocation (128K of the 512K total SRAM is reserved for RTU use). Some LP-1 RTUs and all LP-2/3 RTUs are fitted with 4 MB of expanded flash memory. This expanded flash memory can be used for storing event logs by ticking the *LP1/2/3 Exp. Memory* box. When ticked (enabled), all event logs are stored in flash memory only (SRAM is not used). Note: event logs stored in expanded memory can only be uploaded using Toolbox 32. They cannot be uploaded using the Citect or Wonderware Kingfisher driver.

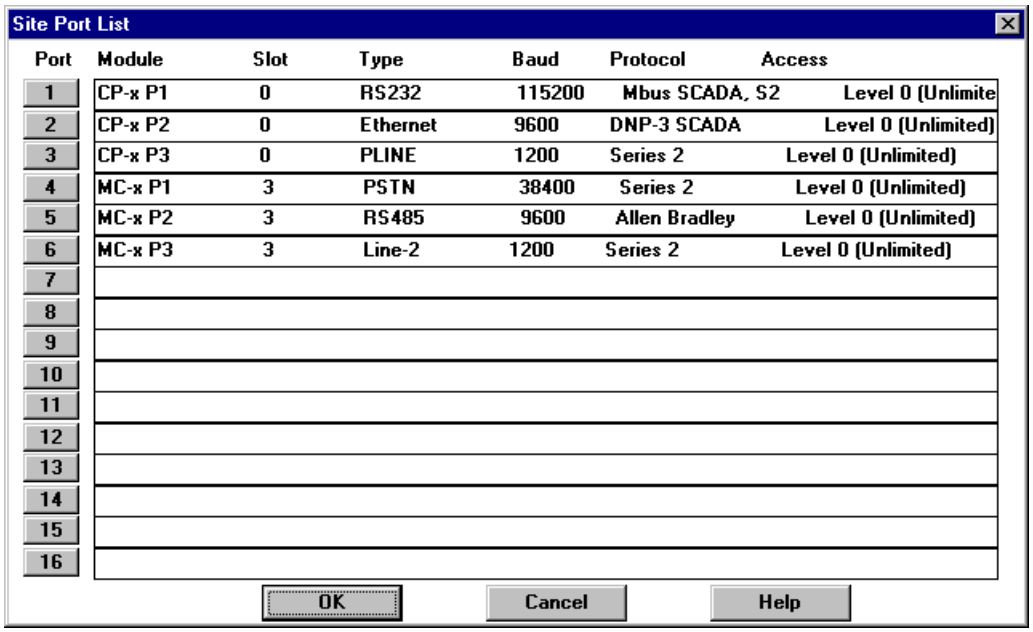
Store ladder logic (.LL) files in RTU: When ticked, every time ladder logic is downloaded to the RTU, the compiled logic and the ladder source file will both be downloaded. This ensures the latest copy of the ladder source file is always stored in the RTU. Future changes can be made by uploading the ladder source file from the RTU using Toolbox**,** performing the changes, compiling the ladder logic and then downloading the compiled logic and the ladder source file again. Memory space is automatically allocated for the ladder source file by Toolbox. (To determine the amount of memory that will be allocated, simply view the size of the ladder source file (*FILENAME.LL*) on the PC using Windows Explorer).

Check Memory Usage: Provides details about the memory usage of the RTU configuration file. Will also attempt to communicate with the RTU and display the available memory in the RTU. Eg:

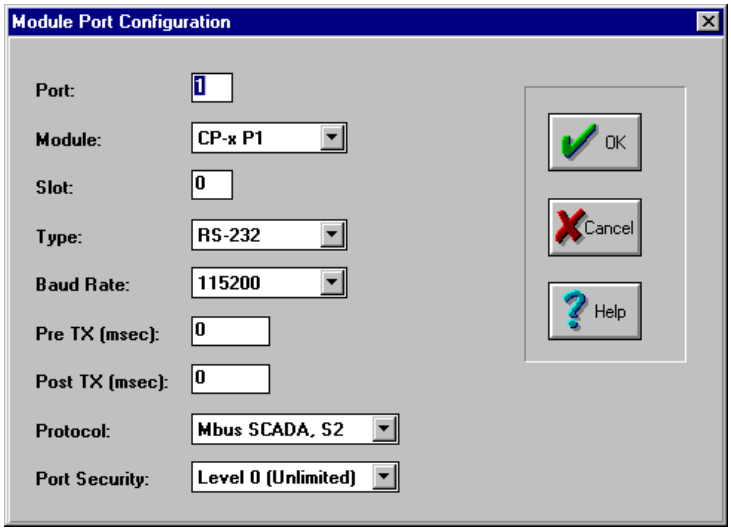

Check RTU Memory Usage
RTU Processor Type : CP-21
RAM Size : 2048 kB (user memory = 1344 kB)
Firmware Version : 1.44a
Memory Usage
Event Logs : 64 kB (max number of logs = 5461)
Network Registers : 64 kB (max network blocks = 496)
Picture Buffer : 0 kB
Firmware Drivers : 0 kB
Compiled Logic : 64 kB
Ladder Logic (.LL) files : 1 kB
Phone Directory : 0 kB (0 phone directory entries)
DNP3 Comms Ports : 0 kB
Total Memory Usage : 193 kB
Available Memory : 1151 kB
OK
Cancel
Help

Configuration - Port List

Defines the settings for each of the 16 possible communication ports of an RTU.


Site Port List
Port Module Slot Type Baud Protocol Access
1 CP-x P1 0 RS232 115200 Mbus SCADA, S2 Level 0 (Unlimite
2 CP-x P2 0 Ethernet 9600 DNP-3 SCADA Level 0 (Unlimited)
3 CP-x P3 0 PLINE 1200 Series 2 Level 0 (Unlimited)
4 MC-x P1 3 PSTN 38400 Series 2 Level 0 (Unlimited)
5 MC-x P2 3 RS485 9600 Allen Bradley Level 0 (Unlimited)
6 MC-x P3 3 Line-2 1200 Series 2 Level 0 (Unlimited)
7
8
9
10
11
12
13
14
15
16
OK
Cancel
Help

The settings for each port can be configured by selecting the **Port** button (**1** to **16**).



Port: Not configurable.

Module: The module type and the port number to configure. A PC-1 has one fixed RS232 port and one option port. A CP-10/11/12/21 or MC-10/11/12 has one fixed RS232 port and two option ports. An LP-1/2/3 has two fixed serial ports and one (LP-2) or two (LP-1/3) option ports. Possible Module settings:

- **NONE:** Clears existing port configuration
- **CP-x P1:** CP-10/11/12/21 or LP-1/2/3 port 1
- **CP-x P2:** CP-10/11/12/21 or LP-1/2/3 port 2
- **CP-x P3:** CP-10/11/12/21 or LP-1/3 port 3
- **CP-x P4:** LP-1/2/3 port 4
- **PC-1 P1, PC-1 P2:** PC-1 ports 1 and 2 respectively
- **MC-x P1, MC-x P2, MC-x P3:** MC-10/11/12 ports 1 to 3 respectively
- **RD-1, LM-1:** (Superseded)
- **LM-2 P1, LM-2 P2:** (Superseded)

Slot: The slot address in which the module is positioned (1 to 64). It is not necessary to specify a slot address for a CP-xx or a PC-1 module.

Type: The type of the port as labelled on the RTU module. Option boards installed on CP-xx modules are displayed from the Hardware Overview as *None* (no option board installed), *UART Detected* (SER-S, SER-I, V34-D, Line-2, Fibre or HART option board), *PSTN Modem* (V22-d option board) *Ethernet* (E'NET-E or E'NET option board), *Line Modem / Radio* (LINE-L option board) or *Image* (IMAGE-J option board). The Type setting should match the option board type as detailed below:

- **RS232:** RS232 serial communications with RTS-CTS control. This setting is also used for fibre optic ports and for spread spectrum radio ports if the radio does not need to be configured with an initialisation string. A fibre optic port is configured the same way as an RTU serial port.
- **RS485:** RS485 serial communications (used for multiple RTUs connected to a two-wire highway up to 600 metres long). Please see the topic Example - RS485.
- **RS422:** RS422 serial communications (used for multiple RTUs connected to a four-wire highway). Please ensure Pre TX and Post TX are configured as detailed below.
- **RADIO:** Original radio connection for a PC-1/MC-1 Radio option boards or for CP-xx/MC-xx LINE-L option boards. Note: CP-xx and MC-xx ports operate at 1200 baud while PC-1 and MC-1 ports operate at 300 or 1200 baud. LINE-2 option boards are configured as *Line-2* (please see below) instead of RADIO. The output power of a radio port can be configured by selecting the **Configure** button as shown in Configuration - RADIO/PLINE. Please see the topic Example - Radios.
- **PLINE:** Private line connection for PC-1/MC-1 Private Line option boards or for CP-xx/MC-xx LINE-L option boards or LM-2 ports. Note: CP-xx/MC-xx and LM-2 ports operate at 1200 baud while PC-1/MC-1 ports operate at 300 or 1200 baud. LINE-2 option boards are configured as *Line-2* (please see below). The output power of a private line port can be configured by selecting the **Configure** button as shown in Configuration - RADIO/PLINE. Please see the topic Example - Private Line.
- **PSTN:** For external dial-up modems, Dial option boards, GSM modems and GPRS modems. PSTN can be configured on any RS232 port and has a number of configurable parameters that are configured by selecting the **Configure** button as shown in Configuration - PSTN. Please see the topic Example - Modems.
- **TMR:** Trunk Mobile Radio communications. Uses RS232 serial communications and a special form of message encoding. To be able to use the TMR protocol a special version of RTU firmware is required. TMR has a couple of configurable parameters that can be configured by selecting the **Configure** button as shown in Configuration - TMR.
- **RADIO HOT:** (For PC-1/MC-1 only) For use with FSK radios that constantly transmit or receive a carrier signal. When this option is selected, the receiving RTU does not wait for the carrier to drop before replying. The receiving RTU ignores the Quiet Time Before TX System Parameter and replies immediately. Please see *Line-2 HOT* for Line-2 option boards.
- **Mobitex:** For Mobitex radio communications on one of the RTU's serial ports. A firmware driver is required in order to use the Mobitex protocol.
- **Ethernet:** The Ethernet port settings can be configured by selecting the **Configure** button as shown in Configuration - Ethernet. Ethernet can be used on CP-10/11/12 ports 2 and 3, MC-10/11/12 ports 2 and 3 or CP-21 port 2. CP-21 Ethernet supports the Kingfisher protocol all the time AND *one* of the following protocols: Modbus or DNP3. CP-10/11/12 Ethernet supports the Kingfisher protocol all the time AND *one* of the following protocols: Modbus, DNP3 or Allen Bradley. MC-10/11/12 Ethernet only supports the Kingfisher protocol.
- **Image Capture:** Image capture option board. When this option is selected, settings for Baudrate, Pre TX, Post TX and Protocol are ignored. Please see the topic Example - Kingfisher Images.
- **RS232 RADIO:** For radios with an RS232 interface and a radio frequency (RF) Carrier Detect signal. Carrier detect is used to prevent messages being sent when the radio network is busy. RS232 Radio also uses the Pre TX and Post TX port settings and allows a carrier test to be performed on the port. Please see the topic Example - Radios.
- **LP-1 FFSK P3:** [Superseded!] For LP-1 port 3 integrated radio. The integrated radio has a number of configurable parameters that can be configured by selecting the **Configure** button as shown in Configuration - LP-1 Integrated Radio.
- **LP-1 FSK P3:** For LP-1 FSK port 3 Trio radio (TC-MR450). Please ensure Pre TX and Post TX are configured as detailed below.
- **GPS Internal:** GPS option board for CP-21. Please see the topic Driver - GPS.
- **GPS External:** External GPS device.
- **Line-2:** Private Line or radio connection for Line-2 option boards. Please see the topic Example - Private Line or Example - Radios.
- **Line-2 HOT:** Same as RADIO HOT except applies to Line-2 option boards.

- **GPRS:** General Packet Radio Services. Allows connection to an original GPRS modem when using the GPRS firmware driver. Please see the topic Example - GPRS Modems.
- **SS Radio:** Spread spectrum radio option board or external radio. Allows an initialisation string to be set to the radio by selecting the **Configure** button as shown in the topic Configuration - Spread Spectrum Radio. To obtain radio data or for other port settings please see the topic Driver - Spread Spectrum Radio. Note: RS232 can be used for the port type if the radio does not need to be configured.
- **GPRS2:** General Packet Radio Services. Allows connection to an updated GPRS modem when using the GPRS2 firmware driver. Please see the topic Example - GPRS Modems.

Baud rate: (300 to 115200) The speed at which the RTU will send or receive messages. Note: Radio and Private Line ports have a maximum baud rate of 1200. When using a PSTN modem to dial a paging service, set the Baud rate to 9600 (or 2400 if experiencing problems connecting reliably). An LP-1/2/3 has a maximum baudrate of 38400 on ports 1 and 2.

Pre TX: For radio, private line, PSTN and Ethernet ports.

- Radio and private line ports: Pre Tx defines how long the carrier and RTS are transmitted for before data is sent. Radios commonly require a Pre TX of 300ms while private lines use 50 to 100ms. 10ms is used for RS485 and RS422.
- PSTN ports: the RTU will wait a minimum of 3 seconds after a carrier is detected before setting the online bit. Pre TX can be used to set the amount of extra time the RTU will wait after a carrier is detected before setting the online bit and allowing messages to be initiated. A setting of 0 is used for most PSTN modems or set to 20,000 ms for a Motorola 9522 satellite phone.

Post TX: For radio, private line, PSTN and Ethernet ports.

- Radio and private line ports: Post TX defines how long the carrier and RTS are transmitted for after the data has been sent. Radios commonly require a Post TX of 100ms while 10 to 50ms is used for private lines. CP-xx line option boards require 50ms Post TX when used with private lines. RS485 and RS422 sometimes require very short Post TX delays. The following values can be used for CP-10/11 RTUs:
  0 = 0-5 ms (5 ms resolution)
  1 = 0-1 ms (1 ms resolution. This is the shortest Post TX setting)
  2 = 1-2 ms (1 ms resolution)
  3 = 2-3 ms (1 ms resolution)
  4 = 3-4 ms (1 ms resolution)
  5 = 4-5 ms (1 ms resolution)
  All other settings (6-65535) have a 5 ms resolution
- PSTN ports: Post TX is the maximum delay before the RTU decides the message being received has finished. Set to 0 for most PSTN modems or set to 400 ms for 3G modems. Post TX is used to allow for breaks in the message received (as occurs with the 3G network). If Post TX is set below 350 ms, the RTU will use a minimum value of 350ms.
- Ethernet ports: The inactivity timeout for each Ethernet socket. Once the socket has been inactive for this time setting, the socket will be closed. Implemented in CP-10/11 firmware 1.43d, CP-21 firmware 1.45e and MC-11 firmware C153 or newer. All CP-12/MC-12 has this feature implemented.

CP-21 Ethernet		**CP-10/11/12, MC-10/11/12 Ethernet**	
Post Tx	Socket Timeout (seconds)	Post Tx	Socket Timeout (seconds)
0	60,000 [default]	0	60 [default]
1 to 9	10	1 to 600	1 to 600
10 to 600	10 to 600	> 600	600
> 600	600		

Protocol: The protocol that the RTU is able to use on the port. Protocols require a firmware driver* to be loaded in the RTU and cannot be used with other protocols on the same port (with the exception of Modbus and Kingfisher). The available protocols are:

- **Series 2:** Kingfisher Series 2 protocol. This is the default Protocol setting.
- **ADS Data Logger:** Device protocol.
- **Allen Bradley:** Allen Bradley PLC.
- **ALERT**: Radio reporting gauge.
- **Alstrom Relay:** Device protocol.
- **ASCII No Parity:** Device protocol.
- **ASCII Even Parity:** Device protocol. Same as above but with even parity.
- **BCL ARC Device:** Device protocol.
- **Conitel:** Device Protocol
- **Cooper:** Device protocol.
- **DATAC:** DATAC RTU.
- **Datataker:** Device protocol.
- **Datran DT300:** Device protocol.
- **DNP-3:** Please see the driver documentation from Semaphore.
- **DV1000:** Device protocol.
- **Form 4C:** Form 4C Reclosure.
- **FUJI Micrex-F:** Fuji Micrex-F PLC.
- **FUJI NJ Series:** Device protocol.
- **GE CCM:** Device protocol.
- **GE SNP-X:** Device protocol.
- **GE T60 Relay:** Device protocol.
- **Genisys:** Genisys train controller (slave).
- **Genisys Master:** Genisys master train controller.
- **GPS:** CP-21 GPS option board (superseded).
- **GPS NMEA:** NMEA 0183 ASCII protocol. For GPS devices connected to an RTU serial port. Eg. GM-66-232 Mini GPS Locator.
- **Hart:** Device protocol.
- **IDEC PLC:** Device protocol.
- **INLINE, INLINE2:** Inline flow computer.
- **INTRAC:** Motorola INTRAC RTU.
- **JZA Train Control:** Device protocol.
- **MAC-800:** MAC 800 RTU.
- **Mercury:** Mercury flow computer.
- **Microtran:** Email Microtran RTU.
- **Mier Transposer:** Device protocol.
- **Minitran:** Email Minitran RTU.
- **G&F Minitran:** Gas and Fuel Minitran RTU.
- **MTran Host:** Minitran Host, South East Water.
- **Mbus ASCII Init** [#]**:** Modbus protocol for an MC-xx port only. Identical to *Mbus init, S2* but uses ASCII data format instead of Modbus RTU format.
- **Mbus init, S2** [#]**:** Modbus and Series 2 protocols. Allows the RTU to initiate Modbus messages using ladder logic and to relay output messages in Modbus format.
- **Mbus init&resp, S2** [#]**:** For CPU ports only. Modbus and Series 2 protocols. Allows the RTU to initiate its own Modbus messages using ladder logic and respond to Modbus messages from other devices. This option should only be selected if two devices are connected using the Modbus protocol and both sides will initiate messages.
- **Mbus SCADA, S2** [#]**:** Modbus master and Series 2 protocols. The RTU will respond to Modbus messages that are for itself and for any other RTU. This option is used for the master RTU which stores copies of the outstation RTU data. If data is requested from an outstation RTU, the RTU will respond

with its own network data corresponding to that RTU. The local RTU will also relay Modbus output messages in either Series 1, Series 2 or Modbus format. Modbus messages are relayed in Series 1 format to outstations that have a system ID of AC or in Series 2 format to outstations that have a system ID of AE. Modbus messages are relayed in Modbus format if the port that the RTU relays the message out of is configured as a Modbus initiating port. Note: after successfully relaying an output message, the network register corresponding to the output will also be updated in the local RTU.

- **Mbus slave, S2** [#]**:** Modbus and Series 2 protocols. The RTU will respond to Modbus messages that are for itself only.
- **Mbus init&parity** [#]**:** Identical to *Mbus init, S2* but uses an even parity port setting and will not respond to Series 2 messages.
- **Mbus init&resp&par** [#]**:** For CPU ports only. Identical to *Mbus init&resp, S2* but uses an even parity port setting and will not respond to Series 2 messages.
- **Mbus SCADA&parity** [#]**:** Identical to *Mbus SCADA, S2* but uses an even parity port setting and will not respond to Series 2 messages.
- **Mbus slave&parity** [#]**:** Identical to *Mbus slave, S2* but uses an even parity port setting and will not respond to Series 2 messages.
- **Modem Sw Unit:** Email Modem Switch Unit.
- **Monitor WeatherStn:** GLX data logger.
- **Multitrode 2PC:** Multitrode pump controller.
- **NEC DCU:** NEC PLC.
- **OMRON:** Omron PLC.
- **PEEK:** PEEK traffic light controller.
- **Remote Data Logger:** Protocol for Sagasco.
- **S1 Ctrl, Mbus, S2** [#]**:** Series 1 Controller, Modbus and Series 2 protocols. Identical to *Mbus SCADA, S2* but will also respond as a Series 1 master RTU. Will also relay series 1 messages to RTUs that are in the Network List.
- **S1 Outstn, S2:** Series 1 Outstation and Series 2 protocols. Will not relay series 1 messages for other RTUs.
- **S1 Ctrl, no CRC, S2:** Original Series 1 Controller (for use with CPU1 modules) and Series 2 protocols. Will relay series 1 messages to RTUs that are in the Network List.
- **S1 Out, no CRC, S2:** Original Series 1 Outstation (for use with CPU1 modules) and Series 2 protocols. Will not relay series 1 messages for other RTUs.
- **SER_SSPA:** Device protocol.
- **Shaft Encoder:** Hohner Shaft Encoder.
- **Simatic TI:** Siemens Simatic 500 series PLCs..
- **SM6615:** Device protocol.
- **STIC Gauge:** Enraf STIC receiver.
- **Tandberg:** Digital receiver and decoder.
- **Traffic Light Controller:** Device protocol.
- **TRIO Eseries:** Trio E series radio interface.
- **TS5000:** TS5000 RTU.
- **User-Defined:** User defined protocol configured in ladder logic. When *User-Defined* is selected, another configuration box will appear. The number of data bits (7 or 8), the parity (none, even or odd) and the number of stop bits (1 or 2) can then be specified.
- **YSI Logger:** Device protocol.

Port Security: (Level 0 to 5) The security access level of each port. Port security is enabled by loading the Security driver and configuring the port with a non-zero security level. Please see the appendix - RTU Security for more information.

* All protocols except **Series 2** require a special version of firmware or a firmware driver. Special versions of firmware and firmware drivers are available from Semaphore. Standard firmware drivers are available from http://helpdesk.servelec-semaphore.com. Some protocols will operate on both PC-1/CP-xx and MC ports while other protocols operate on PC-1/CP-xx ports only or MC ports only as detailed in **Protocols.pdf** available from http://helpdesk.servelec-semaphore.com.

[#] Note 1: When using the Modbus port protocols, some addresses cannot be used for external Modbus devices. These addresses correspond to the SYNC characters used for the Series 2 and Series 1 protocols. Since Modbus messages begin with the Modbus device address, these messages can be confused with Series 2 and Series 1 messages which begin with sync characters AE (Series 2), AC (Series 1 CPU3) or A5 (Series 1 CPU1). Each Modbus protocol port setting will also respond to Series 2 messages and so Modbus device address 174 (AE Hex) cannot be used. If the Series 1 CPU3 port protocol is used, then Modbus device address 172 (AC) cannot be used. If the Series 1 CPU1 port protocol is used, then Modbus device address 165 (A5) cannot be used.

Configuration - PSTN

If the port type is set to **PSTN**, selecting the **Configure** button will display the following window. Note: the first 4 parameters also apply to GPRS modems.

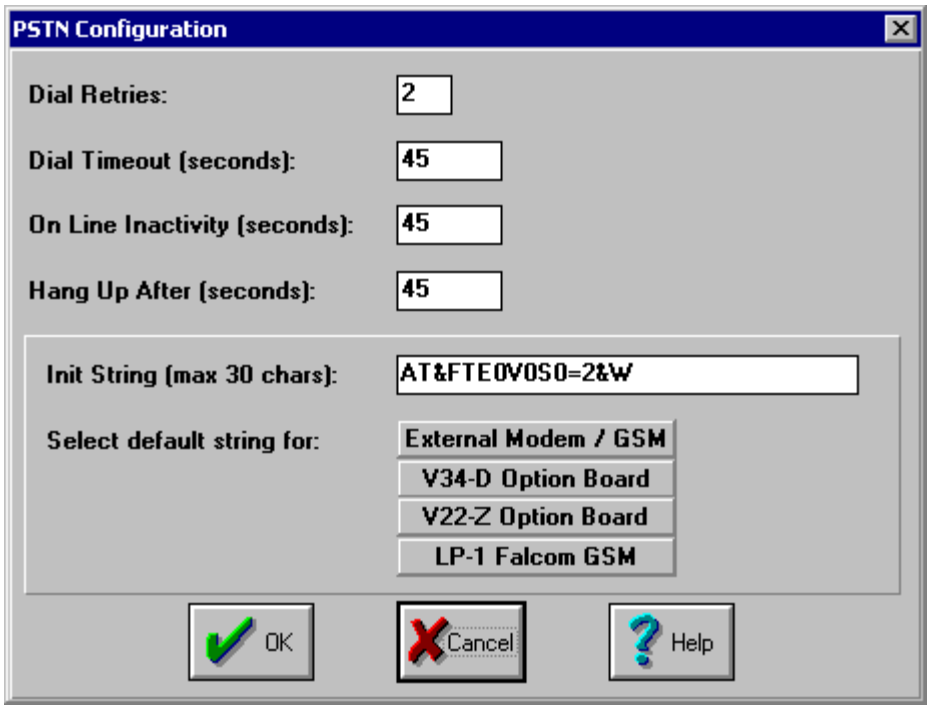


Figure: Default PSTN Configuration Window showing Initialisation String for an External Modem or GSM.

Dial Retries: The number of times that the RTU will attempt to re-dial the target RTU. The time waited before performing the next dialing attempt changes according to the dialing attempt number. After the first dialing attempt, the RTU will wait 30 seconds and then dial again. The RTU then increments the wait time by 1 minute between subsequent dialing attempts.

After	RTU Waits:
1st Dial	30 seconds
2nd Dial	1 minute 30 seconds
3rd Dial	2 minutes 30 seconds
4th Dial	3 minutes 30 seconds etc

Dial Timeout (seconds): The time from when dialing begins that is waited to get carrier detect (carrier detect occurs a short time after the receiving modem has answered). When dialing a GSM, the Dial Timeout should be set to at least 45 seconds.

Automatically Hanging Up
An RTU can be configured to automatically hang up after a certain amount of time has elapsed from *sending* the last message or from *receiving* the last message. There are two parameters used to specify these times and the parameter that is used is dependent on whether the last message was transmitted or received. If the last message was transmitted, the RTU will wait for *Hang Up After* seconds and then hang up. If the last message was received, the RTU will wait for *On Line Inactivity* seconds and then hang up.

On Line Inactivity (seconds): (0-32767) The RTU will hang up after this amount of time has elapsed since the last message *received*. A value of 0 disables the function.

Hang Up After (seconds): (0-32767) The RTU will hang up after this amount of time has elapsed after connection or after *sending* the last message. A value of 0 disables this function.

Remaining Online

To remain online after connection, set **On Line Inactivity** to 0 and **Hang Up After** to 0 in *both* RTUs in the PSTN link. If the line is disconnected, the RTU will reconnect when the next TX or RX message is initiated from ladder logic.

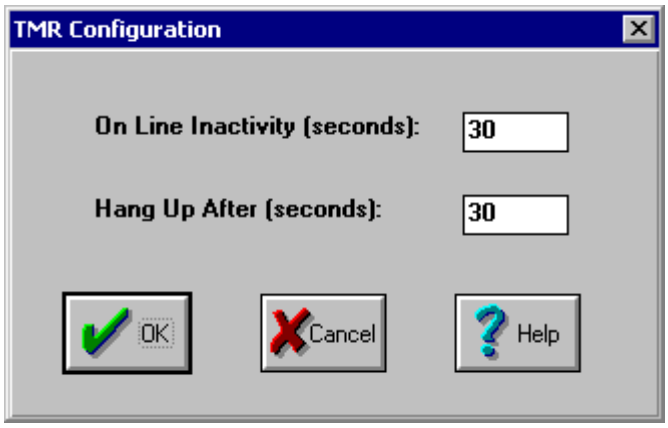
Init String: (0-29 characters) These characters are sent to the modem to initialise it after startup, after disconnection and if a dial attempt fails. The default initialisation strings for each type of modem can be obtained by selecting the appropriate button below the Initialisation String field and are listed below.

Modem	**Default Initialisation String**
3G Modem (Maxon Modmax MM-6280IND)	[blank]. The default string should be removed for the modem to function correctly. Note: modem parameters must be set and saved in the modem using a terminal program. Please see the topic Example - Modems.
External Modem or GSM (eg. Banksia modem or Wavecom GSM) #	AT&FTE0V0S0=2&W
V34-D Option Board #	AT&FTE0V0S0=2&W X3 can be added to the above string if the modem is unable to recognize the dial tone or is experiencing problems establishing a connection ie. AT&FTE0V0S0=2X3&W
V22-D Option Board (superseded)	AT&FTE0V0S0=2&C1&W &C1 = track DCD from remote For this option board, the port baud rate should be set to 2400 bps (or lower).
LP-1 Falcom GSM (superseded)	AT@|+CICB=0|+CBST=7|+DS=0|&W LP-1 interprets the above string as separate AT commands as shown below: ATE0%C0\N0S0=2&D2 <Enter> AT+CICB=0 <Enter> AT+CBST=7 <Enter> AT+DS=0 <Enter> AT&W The @ character is replaced by *E0%C0\N0S0=2&D2* by the LP-1 (this keeps the initialisation string below the 29-character limit). *&D2* allows the RTU to hang up the GSM using DTR. *+CICB=0|+CBST=7|+DS=0* sets the GSM to 9600 baud data mode. The | character is replaced by a carriage return and *AT* by the LP-1.

Dialing A Paging Service: If experiencing problems, error correction may need to be disabled by including \N0 ie. AT&FE0V0S0=2\N0&W. If experiencing problems when using an MC module and a Dial option board, the baudrate may need to be limited to 9600 by including F8 in the initialisation string ie. AT&FE0V0S0=2F8&W. When using a recent Dial option board, the RTU port baudrate may need to be set to 2400 baud. Older Dial option boards can use 9600 bps.

Configuration - TMR

If the port type is set to **TMR** (Trunk Mobile Radio), selecting the **Configure** button will display the following options:



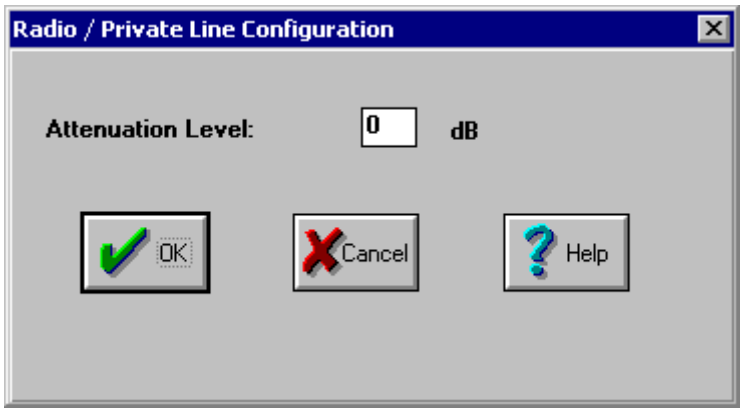
On Line Inactivity: (0-32767) The RTU will hang up after this amount of time has elapsed since the last message *received*. A value of 0 disables the function.

Hang Up After: (0-32767) The RTU will hang up after this amount of time has elapsed after connection or after *sending* the last message. A value of 0 disables this function.

Note: TMR is supported by PC-1 RTUs running a special firmware version. Firmware is available on request.

Configuration – RADIO / PLINE

If the port type is set to **RADIO** or **PLINE**, selecting the **Configure** button will display the following options. Note: attenuation Level only applies to the superseded LINE-L option boards.

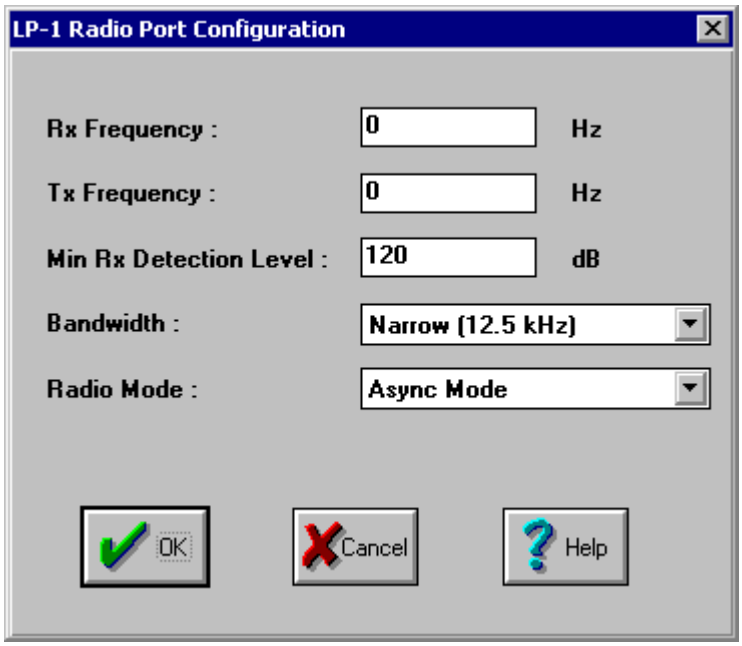


Attenuation Level: (0-15) The TX Audio output level is reduced in power by this amount. The default setting *0* maintains the maximum output power (-6 dBm).

Attenuation Level	Output dBm	Signal level mV RMS (into 2 x 600 Ohm)
0	-6	400
1	-7	350
2	-8	310
3	-9	276
4	-10	251
5	-11	223
6	-12	195
7	-13	173
8	-14	158
9	-15	141
10	-16	123
11	-17	110
12	-18	100
13	-19	89
14	-20	78
15	-21	69

Configuration - LP-1 Integrated Radio

Superseded LP-1 hardware option. If the port type is set to **LP-1 FFSK P3**, selecting the **Configure** button will display the following options:



Rx Frequency: (0-512,000,000 Hz) The frequency of incoming messages.

Tx Frequency: (0-512,000,000 Hz) The frequency of outgoing messages.

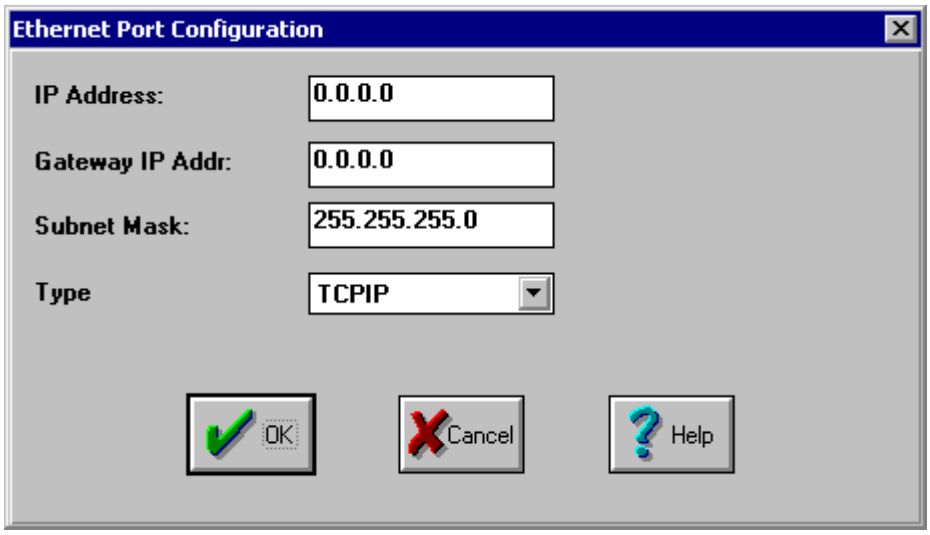
Min Rx Detection Level: (90-120 dB) The weakest detectable signal strength. Note: radio noise is received between -120 to -108 dB. A setting of 90 (ie. -90dB) is recommended. The larger the number, the weaker the signal that can be detected by the radio.

Bandwidth: (12.5, 20 or 25 kHz) Bandwidth of the radio. Standard setting is 25kHz (Wideband).

Radio Mode: (Async Mode, Sync Mode (20bpp), Sync Mode (253bpp)) Standard setting is Sync Mode 20bpp.

Configuration - Ethernet

If the port type is set to **Ethernet**, selecting the **Configure** button will display the following options:



IP Address: A local area network may be used to communicate with an RTU that has an Ethernet option board. Each number in the IP address can have values in the range of 0-255. The IP address can then be used by Toolbox to communicate with the RTU via Ethernet.

Gateway IP Addr: The IP address that allows access to the outside world for communications to an RTU on another local area network. Each number in the Gateway IP address can have values in the range of 0-255.

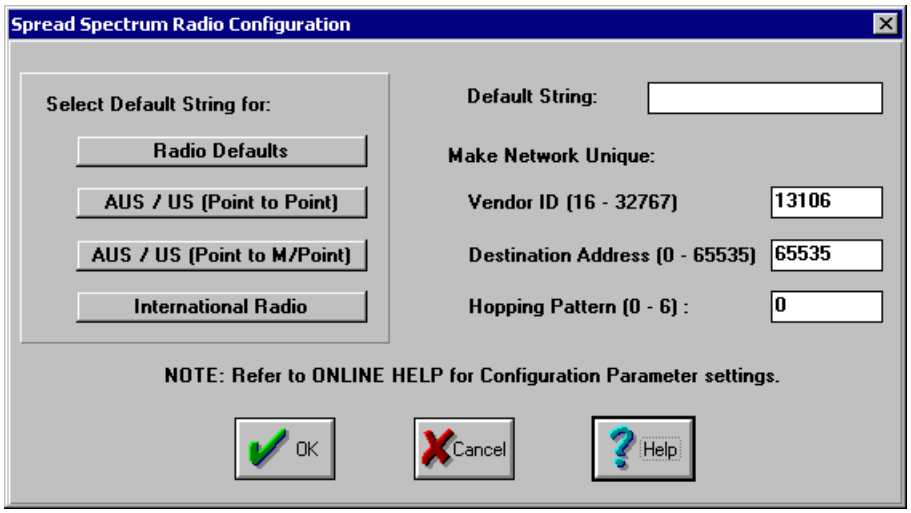
Subnet Mask: Allows for detailed configuration of devices on a LAN. The default setting of 255.255.255.0 is used in most cases. Each number in the Subnet Mask can have values in the range of 0-255.

Type: (TCPIP or UDPIP) Ethernet transport layer to use.

For more information about Ethernet ports, please see the Ethernet appendix.

Configuration - Spread Spectrum Radio

If the port type is set to **SS_RADIO**, selecting the **Configure** button will display the following options:



The spread spectrum radio uses extended AT commands to set or read various parameters.

Default String: The beginning of the initialisation string sent to the radio. When configured (ie the Default String is not blank), the RTU automatically adds the Vendor ID, Destination Address and Hopping Pattern commands to the end of the string and then sends the complete string to the radio in the format:

[Default String],[Vendor ID],[Destination Address],[Hopping Pattern],WR

Default Strings:

Radio Defaults	<blank - no string sent to radio>
AUS/US (Point to Point):	ATBR0,MT0
AUS/US (Point to Multi-point):	ATBR0,MT2
International Radio	AT

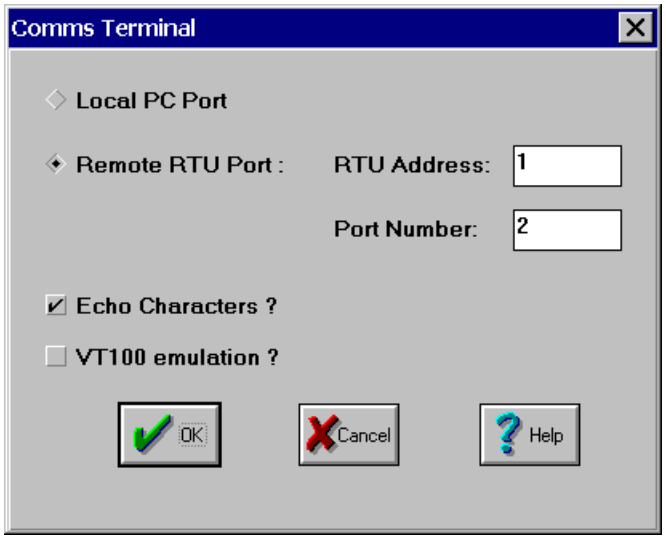
BR0: Data is transmitted between the radios at 9600 bps. Note: if a higher baud rate is used, the transmission range may be reduced.
MT: Multi-Transmit mode (Australia / US only. Not required for the International radio). MT0 [default] = point to point mode, MT2 = point to multi-point mode.

Vendor ID: (16-32767, default=13106) Sets the ID number of the Spread Spectrum radio. All radios on the same network need to have the same Vendor ID in order to communicate with each other. It is recommended that Vendor ID be changed to avoid interference with other radio networks. A setting of 13106 can be used to communicate with G3 spread spectrum radios (by default).

Destination Address: (0-65535, default=65535) Sets the Destination address of the Spread Spectrum radio. All radios on the same network need to have the same Destination Address in order to communicate with each other. It is recommended that Destination Address be changed to avoid interference with other radio networks. A setting of 65535 must be used to communicate with G3 spread spectrum radios.

Hopping Pattern: (0-6, default=0) Sets the Hopping channel of the Spread Spectrum radio. All radios on the same network need to have the same Hopping channel to enable them to communicate. To minimise interference from another RTU using a spread spectrum radio, a hopping channel number that is different to the offending radio should be used.

Advanced Radio Configuration
A spread spectrum radio can be initialised using the Toolbox Comms Terminal. Eg. for RTU1 with an SS radio option board on port 2, the following Comms Terminal settings would be used:



Once Comms Terminal has started, the radio may first need to be put into command mode by sending +++ <wait for a couple of seconds until OK appears>. Commands can then be sent to the radio. Command examples:

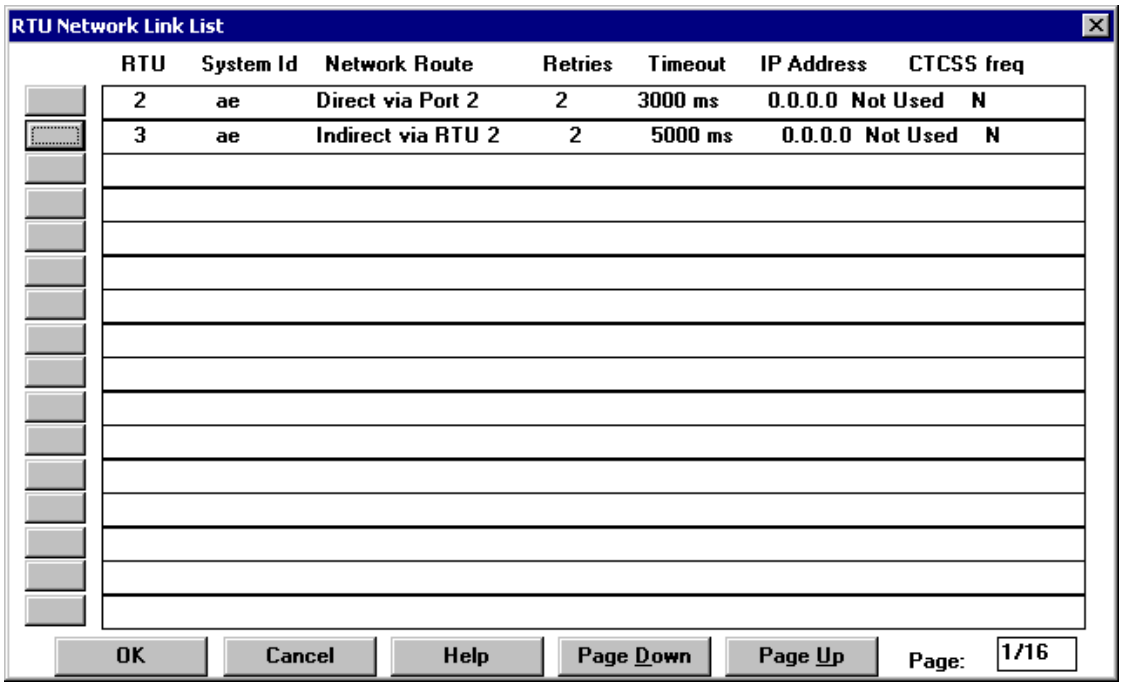
ATRE	Reset to factory defaults
ATPL2	Set Transmit power level to 100mW [default is 1W]
ATDTxxxx	Sets the Destination Address to xxxx hexadecimal
ATWR	Save settings to radio memory
ATID	Read back the Vendor ID setting in the radio

For a comprehensive list of AT commands, a product manual (for the 9Xtend 900MHz OEM RF Module) is available from the supplier - www.digi.com

RTU Port / Network Settings
Please see the topic Driver - Spread Spectrum Radios for port and network link settings. The topic also describes the diagnostic data that is available from the spread spectrum radio itself.

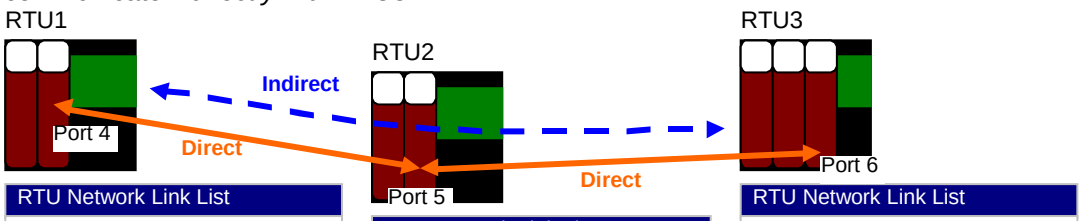
Configuration - Network List

The Network List tells the local RTU how to communicate with remote RTUs.


RTU Network Link List
RTU System Id Network Route Retries Timeout IP Address CTCSS freq
2 ae Direct via Port 2 2 3000 ms 0.0.0.0 Not Used N
3 ae Indirect via RTU 2 2 5000 ms 0.0.0.0 Not Used N
OK
Cancel
Help
Page Down
Page Up
Page:
1/16

By default, an RTU will automatically create or update links in its network list if a message is received that is new or is different to an existing link. Eg. if an RTU has a network link to RTU2 using port 2 and a message is received from RTU2 on port 3, the network link will be updated to port 3. This feature can be disabled using ladder logic (by setting #YMODE.2=1)

An RTU can have *direct* and *indirect* links to remote RTUs. For the example below, RTU1 has a direct communication link to RTU2 and RTU2 has a direct communication link to RTU1 and to RTU3. RTU1 can communicate *indirectly* with RTU3.



RTU Network Link List

Target RTU	Network Route
2	Direct via Port 4
3	Indirect via RTU 2

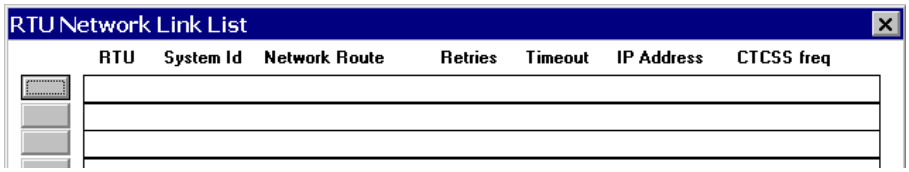
RTU Network Link List

Target RTU	Network Route
1	Direct via Port 5
3	Direct via Port 5

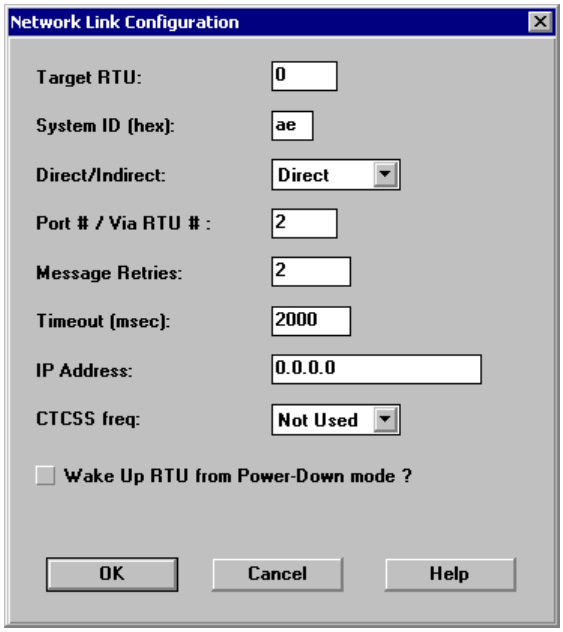
RTU Network Link List

Target RTU	Network Route
1	Indirect via RTU 2
2	Direct via Port 6

A blank network list is shown below.



A network link can be added by selecting the button at the start of a blank row. After selecting a button, a window will appear as shown below.



Target RTU: (0-249) The address of the RTU to communicate with.

System ID: This is the communications sync character used at the start of outgoing Kingfisher messages. An RTU will only respond to Kingfisher messages that begin with the same System ID as the RTU's own system ID (as configured in Configuration, System Parameters) or that begin with AE (Hexadecimal). System ID is not used by other protocols.

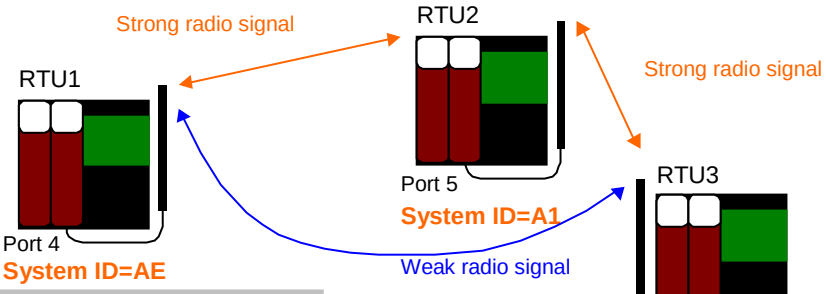
Default system IDs for Kingfisher messages are as follows:

- **AE** - Default for Series 2 RTUs. It is recommended that AE be used for all Series 2 RTUs except Store and Forward RTUs as illustrated below.
- **AC** - Series 1 CPU3 or Series 1 CPU1 with version 3 EPROM
- **A5** - Series 1 CPU1 with version 2 or prior EPROM

A Series 2 RTU knows it is communicating with a Series 1 RTU whenever a system ID of AC or A5 is used. When a Series 2 RTU receives a Modbus output command for a Series 1 outstation, it will convert the message into Series 1 format. Note: A Series 2 RTU will not convert Citect Kingfisher driver outputs into Series 1 format for Series 1 outstation RTUs.

To prevent both RTU2 and RTU3 responding at the same time, RTU2 and RTU3 are configured with unique system IDs as shown below. RTU1 sends the indirect message to RTU2 with a System ID of **A1**. RTU3 will only respond to messages with a system ID of **A2** and so ignores the message. When RTU2 forwards the message to RTU3, the message is sent with a System ID of **A2**. RTU3 then responds to the message. Note: System IDs 00 and FF are reserved and should not be used.

Target RTU	System ID	Route
1	AE	Direct via Port 5
3	A2	Direct via Port 5



Target RTU	System ID	Route
2	A1	Direct via Port 4
3	A2	Indirect via RTU 2

Target RTU	System ID	Route
1	AE	Indirect via RTU 2
2	A1	Direct via Port 6

Direct/Indirect: Direct means the RTU is directly connected to the target RTU (eg. via a private line or radio link). Indirect means the RTU must communicate via one or more other RTUs (called store-and-forward RTUs) to access the target RTU. When an RTU hears a message that is not for itself, it will store and forward (or relay) the message to the destination RTU if it has a network link to that destination RTU.

Port # / Via RTU #: For a Direct connection, this is the local port number (as configured in the Port List) to be used to communicate with the target RTU. For an Indirect connection, this is the directly-connected RTU address via which the message must be sent to reach the target RTU.

Message Retries: The number of times the RTU will retry sending a message to the target RTU if the previous attempts have failed. The maximum number of attempts is one more than the *Message Retries* setting.

Timeout: The time that an RTU will wait for a reply to its first message. If Message Retries is set to 1 or greater, the RTU will try sending the message again. The time waited for a reply to all other message attempts depends on how many message attempts have already been sent and on the Global Timeout setting (as defined in Configuration, System Parameters) as detailed below.

Attempt	Time waited for reply (milliseconds)
1	Timeout
2	Timeout + Global Timeout
3	Timeout + (Global Timeout x (1<Random<2))
4	Timeout + (Global Timeout x (2<Random<4))
5	Timeout + (Global Timeout x (4<Random<8))
etc	

Where *Random* is a random number between the upper and lower limits defined above.
Example: Timeout=4000 ms, Global Timeout=1000 ms. The wait time after each message attempt is therefore in the following ranges:
After attempt 1: Wait = 4000 ms
After attempt 2: Wait = 5000 ms
After attempt 3: 5000 < Wait < 6000 ms
After attempt 4: 6000 < Wait < 8000 ms
After attempt 5: 8000 < Wait < 12000 ms

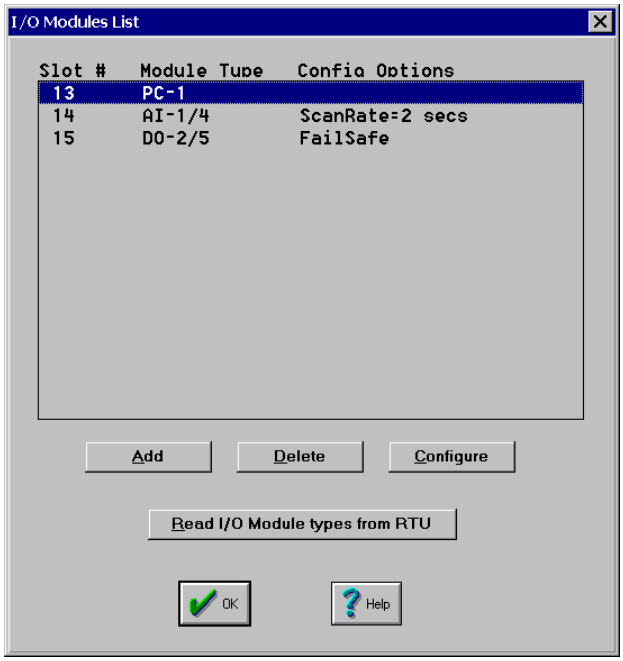
IP Address: (For CP-xx RTUs only) The IP (internet protocol) address of the Target RTU. A local area network may be used to communicate with a CP-xx RTU that has an Ethernet option board. Each number in the IP address can have values in the range of 0-255.

CTCSS freq: This is the frequency to use for encoding outgoing FSK radio messages. CTCSS frequencies can be configured from 67.0 to 250.3 Hz. CTCSS frequencies are used to modulate (encode) a TX signal so that only radios that have the corresponding CTCSS frequency can decode and hear the TX signal. CTCSS encoding is only available with PC-1/MC-1 radio option boards and must be ordered as an extra.

Wake Up RTU From Power-Down Mode? (Only applicable to radio ports) When this option is ticked, the message will be preceded by 4000ms of carrier signal. This will cause a powered-down outstation RTU to wake-up if the outstation RTU port is enabled to wake the RTU up. After waking an RTU up (after the first successful message), subsequent messages do not have 4000ms of Pre-Tx carrier. However, another wakeup message is generated if the local port has been inactive for more than 2 minutes and the last message is not from the powered-down RTU.

Configuration - I/O Modules List

The I/O Modules list is a list of all the modules to be used in the RTU. If the I/O Modules List is configured, Toolbox will check that it matches the modules in the local RTU before downloading the configuration. A warning message will appear if there are any differences. The I/O modules list also allows the various module options to be configured eg. AI-4 scanning rate, DO-2 failsafe outputs or AI-10 input range. If I/O modules are not configured, default settings are used.



To add an I/O module to the list, the slot address of the module must be known or Toolbox can read the I/O modules from the RTU. For a description of slot addresses, please see the appendix - RTU Data, IO Modules.

Add: Adds a module to the list.

Delete: Deletes a module from the list.

Configure: Allows the user to configure the options of the selected module. The various options are detailed in the next section.

Import Module List from RTU: uploads the module list from the RTU.

Configuration - I/O Modules List Options

Scan Priority: (All IO modules) Three priorities are available - High (read IO every scan interval), Medium (read IO once every 3 scan intervals) and Low (read IO once every 10 scan intervals). By default, the scan priority is set to High. For some large RTUs, rapidly scanning all the analog inputs slows down the RTU and is unnecessary. To speed up large RTUs, the analog input modules can be configured to have a Low Scan Priority. Note: a Scan Priority of High is always used by PC-1 RTUs.

Scan Rate: (AI-4 only) The AI-4 module has a configurable scan rate of 1-10 seconds. When not configured, the scan rate defaults to 8 seconds.

Failsafe Outputs? (DO-1, DO-2/5/6, IO-2/3 digital outputs. Not supported by IO-4) If the box is ticked and there is no comms activity between the processor and the IO module on the backplane for 10 seconds, the IO module assumes that the processor has failed and turns off its digital outputs. If not ticked (default), all digital outputs will hold their last value.
Note 1: To use failsafe outputs the IO module must contain I/O code version A17 or newer. This can be determined using the module register #YMVERss=17+ (where ss=slot number) eg. to determine the I/O code version of a module in slot 14, use ladder logic to copy #YMVER14 to a local register. The local register will then contain the code version.
Note 2: Failsafe outputs are not currently implemented for analog outputs. All analog outputs will hold last setting upon failure of backplane communications.

Channel Range: (AI-10 only) The current or voltage input range for all 8 channels. The available options are: (none), 4-20 mA, 0-20 mA [default], +/- 10V**,** +/- 20mA (5V) and +/- 10mA (2.5V). When **(none)** is selected, no configuration is downloaded to the AI-10 module and the default 0-20 mA range is used. Note: in order to use the +/- 10V channel range, the current sensing resistors must be removed from the AI-10.

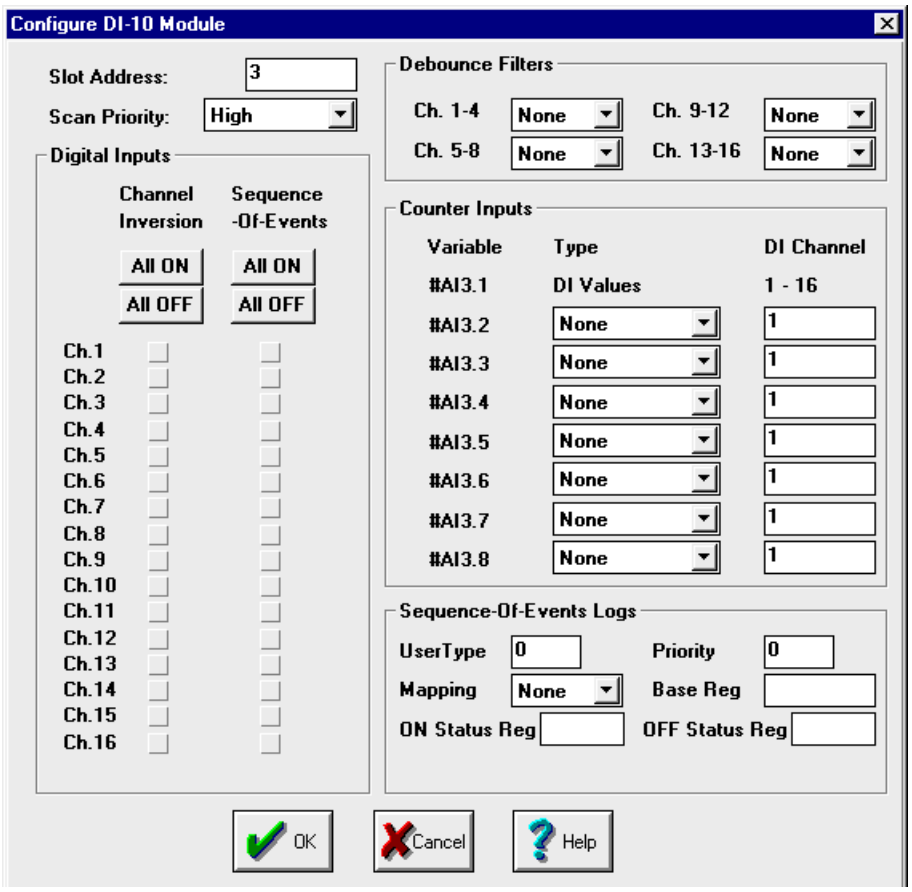
When using the +/- channel ranges, the AI-10 sets bit 16 of the analog input register if the current is negative. If the input goes slightly negative (when floating around 0%), the analog input register will return a very large value (-0.001mA to -20mA =65535 to 32768 respectively). To prevent unnecessary exception reports, bit 16 should be ignored. For the 0-20 mA or 4-20 mA ranges, the input will return zero if the current is negative or below 0 or 4 mA respectively.

Battery Type: (PS-11/21 only) Generic, Sealed Lead-Acid or NiCad. The PS-11 has intelligent battery charging that is varied according to the battery type.

Battery: (PS-11/21 only) 6.5 to 25AH. The size of the battery to charge.

Configuration - DI-10

The DI-10 Module is a 16 channel Digital Input module with configurable Frequency or Pulse or Quadrature counting and Sequence-of-Events recording. After the DI-10 module has been added to the IO modules list, these options can be configured. Note: Sequence of Events is not supported when the DI-10 is used with a PC-1 or CP-1 processor module.



Slot Address: (1 to 64) The slot address of the DI-10.

Channel Inversion: These tick boxes allow channel inversion to be configured for any input channel. The **All ON** and **All OFF** buttons provide an easy way to select or de-select all the channels. By default, a high voltage applied to a digital input channel will result in a logical 1 in the digital input register and the input LED on the module to be set ON. A zero or low voltage level will result in a logical 0 in the digital input register, and the input LED will be OFF. By configuring channel inversion, the situation is reversed, ie. a high voltage results in a logical 0 and the LED is set OFF; a low voltage results in a logical 1 state and the LED is set ON.

Sequence-of-Events: These tick boxes allow Sequence Of Events (SOE) recording to be configured for any input channel. The **All ON** and **All OFF** buttons provide an easy way to select or de-select all the channels. When SOE is enabled, any change of state of the input channel (an event) is logged to 1 millisecond accuracy. This event is automatically included in the Event Log List of the RTU. The DI-10 has a timer that is automatically synchronised with the real-time clock of the processor module. Note: The DI-10 has an internal buffer with enough space for 1000 event logs. This means that a DI-10 can cope with bursts of up to 1000 events at a time. Events are uploaded into the processor module at a maximum rate of 100 events per second allowing the DI-10 to cope with events at a sustained rate of 100 events per second. Events are stored in a circular buffer - which causes the oldest event to be overwritten with the newest event when the buffer is full. Note: memory must be allocated for event logging for SOE to work. Please see the topic Configuration - Memory, Event Logs.

Debounce Filters: (None, 1ms, 3ms, 10ms, 30ms, 100ms, 250ms and AC Filter) These software debounce filters are grouped into configurable filters for channels 1-4, 5-8, 9-12, and 13-16. If a software debounce filter is selected, the logical input in the digital input register will not change to a new state until the actual input has been at the new state continuously for the specified filter time. Note: the sample time for the software debounce filters is in excess of 10 kHz (< 0.1ms between samples). If debounce filtering is selected, this will limit the accuracy of SOE recording on that input channel. *AC Filter* is used when connecting AC inputs to the DI-10 module.

Counter Inputs: (None**,** Freq count, Pulse count or Quad count) The DI-10 has 7 counters that can be configured to count pulses from any of its input channels. Quadrature counting works on pairs of input channels. Channel pairs are 1&2, 3&4, 5&6, 7&8, 9&10, 11&12, 13&14 and 15&16. So selecting Quad Count on channel 1 will actually work with quadrature on channels 1 and 2. Selecting Quad Count on channel 2 will also work with quadrature on channels 1 and 2, but will reverse the phase of the inputs. The same applies to the other channel pairs used for quadrature inputs.

Sequence-Of-Events Logs:
User Type: (1-31) Used to group similar types of logs. For example, analog inputs could be type 1, digital status signals type 2, digital alarm signals type 3 and so on. Logs of one user type can then be uploaded instead of having to upload all the event logs.

Priority: (0-7) Allows separation of logs within each User Type category.

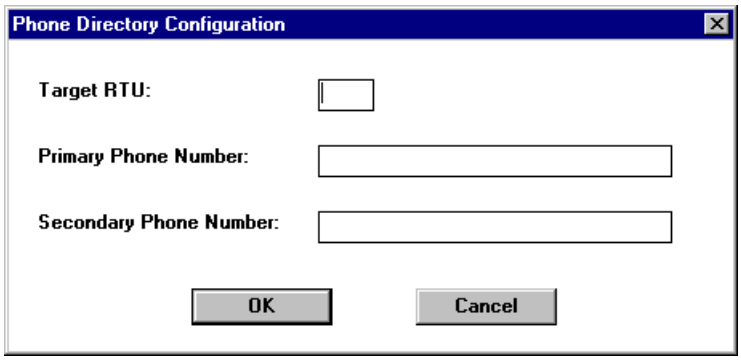
Mapping, Base Reg: (NONE, KF or DNP3) The register name to log when saving the SOE logs. Note: the local registers themselves are not updated with the state of the DI-10 inputs.
NONE: SOE logs are logged as hardware register channels (#DIss.cc). Eg. a SOE log for channel 3 of a DI-10 module in slot 5 would be logged as #DI5.3.
KF: SOE logs are logged as local register bits. The 16 digital inputs are mapped to the 16 register bits of the local register configured in **Base Reg** (if blank, #R1 is used). Eg. if Base Reg = #R100, channel 3 would be logged as #R100.3.
DNP3: SOE logs are logged as local registers. The 16 digital inputs are mapped to 16 consecutive local registers starting at the specified **Base Reg** (if blank, #R1 is used) in DNP3 format. Bit 1 (of 16) in each local register indicates module online and is always set ON. Bit 7 indicates the state of the input. Eg. if Base Reg = #R1025 (first DNP digital input by default), channel 3 would be logged as #R1027. When the digital input is ON, a value of 129 is logged; when the digital input is OFF, a value of 1 is logged.

ON Status Reg, OFF Status Reg: (Blank or #R1 to #R2048) Allows fast rising edge transitions (ON Status Reg) or falling edge transitions (OFF Status Reg) to be detected. When a DI-10 input changes, the corresponding channel in the status register is set ON. These channels can be used to detect momentary alarms that would be missed by ladder logic. Notes:

- The status register channel must be reset using ladder logic (it is not reset by the DI-10).
- The same local register can be specified for *both* the ON and OFF Status Reg. A channel will then be set if there is a change of state of the DI-10 input.
- ON and OFF Status registers will only work for channels that have sequence of events enabled (the channels are triggered from the SOE logs).

Configuration - Phone Directory

The phone directory is used to specify up to 256 phone numbers that an RTU can dial (or up to 512 phone numbers including the secondary phone numbers). By selecting one of the row buttons the window shown below is displayed.



Target RTU: (0-249) The remote RTU that is to be dialed. When using leased line modems (PSTN modems that are hard-wired), the Target RTU must be defined even though a phone number is not dialed.

Primary Phone Number: The number to dial for each odd numbered dialing attempt (ie. dialing attempt 1,3,5...).

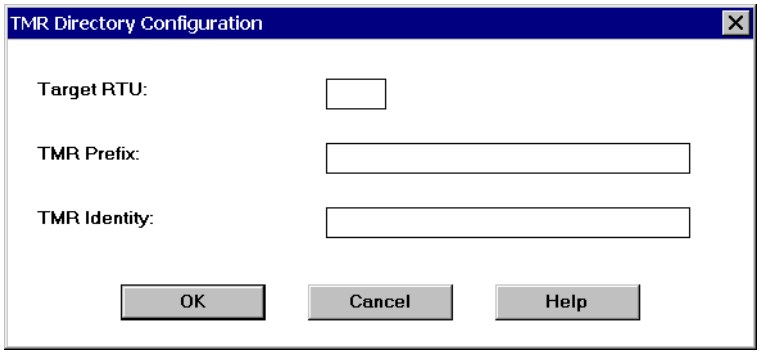
Secondary Phone Number: The number to dial for each even numbered dialing attempt (ie. dialing attempt 2,4,6...). If there is only one phone number to dial for the Target RTU, the Secondary Phone Number should be the same as the Primary Phone Number.

If a **T** (tone dialing) or a **P** (pulse dialing) is used in the Init String of the port list, then this is the default dialing method to be used with all phone numbers. Alternatively, a T or a P can be used as a prefix to the phone number which will cause the RTU to use tone or pulse dialing for that number while ignoring the Init String setting.

Warning: Phone numbers cannot contain spaces

Configuration - TMR Directory

The TMR (Trunk Mobile Radio) directory is similar to the phone directory and allows up to 30 TMR addresses to be configured (for up to 30 RTUs). The protocol used for communicating with a TMR radio is called MAP27 and uses a TMR address comprised of a 7-bit prefix and a 13-bit identity. To be able to use TMR communications, a TMR radio must be connected to one of the CPU's serial ports (port 1 or 2), the port must be configured as type *TMR* and the RTU must be configured with a special version of firmware containing the TMR operating code. The prefix and identity of each TMR radio is available from the TMR supplier.



Target RTU: (1-255) The RTU to communicate with.

TMR Prefix: (0 to 127)

TMR Identity: (0-8191)

Configuration - Pager Configuration

When paging is configured, RTU250 is automatically added to the Network List and is used for the paging communication fail and success counters.

The RTU will send each pager message up to 3 times. Each time the pager message is transmitted, it can be sent to the same group of pager receivers or to a different group of pager receivers. The pager message will not be sent to the next group of pager receivers if the RTU receives an acknowledge within the configured time.

The three groups of pager receivers to send each pager message to are called a sequence. Up to 5 different pager sequences can be defined. When a pager message is configured in ladder logic, the paging sequence to use must be specified. This allows different types of pager messages to be sent to different groups of pager receivers.

A new pager message is sent to the first group of pager receivers in the sequence. If an acknowledgment is not received by the RTU within the first acknowledge time, the pager message is then sent to the second group of pager receivers. If an acknowledgment is not received by the RTU within the second acknowledge time, the pager message is then sent to the third group of pager receivers. If an acknowledgment is not received by the RTU within the third acknowledge time, the RTU will flag a communication fail for RTU250 and increment the fail counter.

Note: to use paging, the RTU must have the paging driver loaded (PAGINGxx.Dxx).

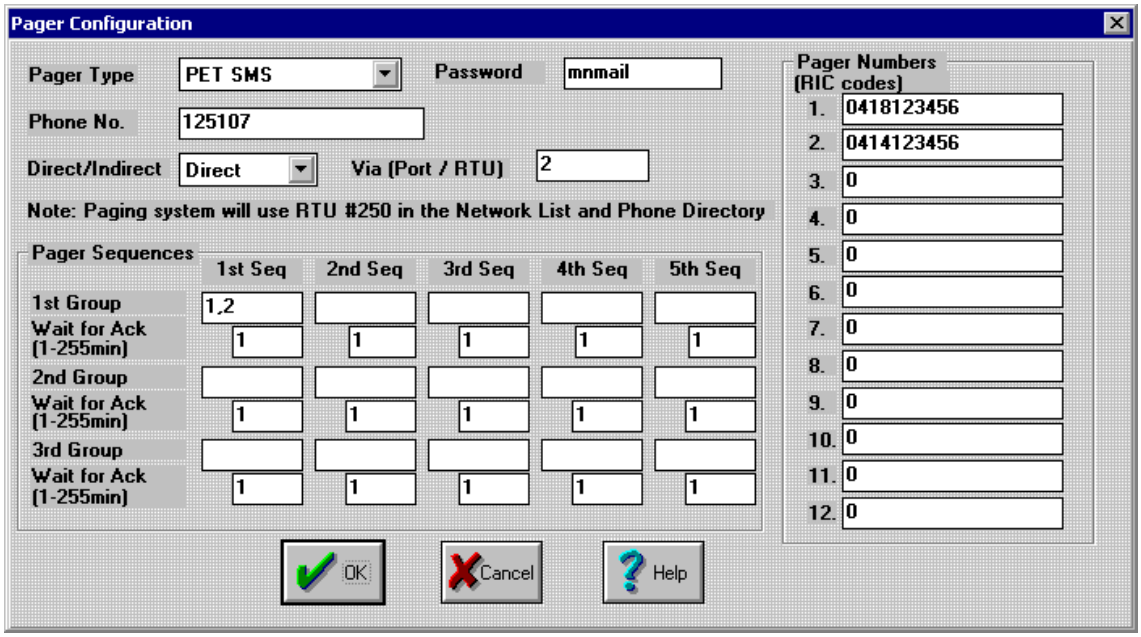


Figure: Pager Configuration Window for Telstra PET SMS paging in Australia.

Pager Type: The type of paging service to be used. The following options are available:

- **None:** Paging is not configured. Selecting *None* prevents the paging parameters from being downloaded into the RTU and avoids Toolbox checking if the RTU contains the paging driver.
- **PET protocol (TAP):** Standard paging protocol.
- **Airtouch:** USA Paging service.
- **Pagenet:** USA Paging service.
- **Radio Pager:** Imark pager transmitter radio.
- **Link PET protocol:** As used by Link Communications and Orange in Australia (8 data bits, no parity, 1 stop bit).
- **POCSAG protocol:** POCSAG pager transmitter radio.
- **GSM SMS:** GSM short message service. For paging from a local GSM directly to a digital mobile phone (does not need to dial a paging service).
- **PET SMS:** PET short message service. For paging to digital mobile phones using a dial-up paging service. As used by Telstra in Australia (7 data bits, even parity, 1 stop bit).

Password: (Optional) Some paging services require a password for validation. Up to 12 characters can be specified. For Telstra PET SMS, **mnmail** can be used for the password. This will allow one message to be sent to *one* Pager Number each time the paging service is dialled. Telstra also supports sending one message to multiple Pager Numbers if an individual password is obtained. To obtain a password, call Telstra on 1800 730 053 and ask about SMS Access Manager.

Phone Number: (Optional) The phone number to dial for the paging service. A Phone Number is not required for a radio pager or GSM SMS. Note: when using a dial-up modem for sending pager messages, any error correction and compression options may need to be disabled (ie. use \N0 in the initialisation string). For Telstra PET SMS, the phone number is 125107 or to dial the service from overseas it is [international dialling code] + 61439125107.

Direct / Indirect: Configure as direct if the pager transmitter is connected to the local RTU or as Indirect if the pager transmitter is connected to another RTU.

Via: The port or the RTU address that the pager transmitter is connected to. Any RTU serial port can be used for paging when using paging driver PAGING11.Dxx or newer.

1st Group, 2nd Group, 3rd Group: This is specified as a local register (ie. #R) or as pager number indexes (ie. 1 to 12) separated by commas (eg. 1,4,7). When a register is used, the lowest 12 channels correspond to the 12 pager receivers respectively. The pager message is sent to each pager receiver that has its channel set ON.

Wait For Ack: The number of minutes to wait for the pager message to be acknowledged (by clearing the acknowledge bit as configured in the pager message ladder block) before transmitting the pager message to the next group of pager receivers. Note: the initiating RTU (the RTU that generates the pager message) can have up to 5 pager messages that are waiting for an acknowledge at any one time. Any additional pager messages will not be sent during this time. Each acknowledge bit will remain ON until manually reset.

Pager Numbers: The pager numbers or RIC codes corresponding to pagers 1 to 12. These can be up to 14 digits long and should all be the same length (requires PAGING11 driver or newer).

Relaying Pager Messages

When a pager message is to be relayed, the initiating RTU will send the text (plus time and date) and the pager number indexes to the pager RTU. This means the RTU that initiates the pager message uses its own pager sequences but not its own pager numbers. The pager numbers configured in the pager RTU (the RTU that dials the paging service or has the pager radio) are used to target the pager receivers. If the pager message is not acknowledged after notifying the first group of pager receivers, the initiating RTU will send a second pager command that targets the second group of pager receivers. To acknowledge the pager message, a zero is written to the acknowledge bit in the initiating RTU (the acknowledge bit is the bit configured in the pager message block in ladder logic).

Fails And Successes

After a pager message is sent to a dial-up paging service, an SMS network or to a pager radio, the RTU does not know if the pager receiver or mobile phone received the pager message (the pager receiver may have been switched off or was out of range). However, with dial-up paging or an SMS network, the RTU is able to test whether the service provider received the message OK.

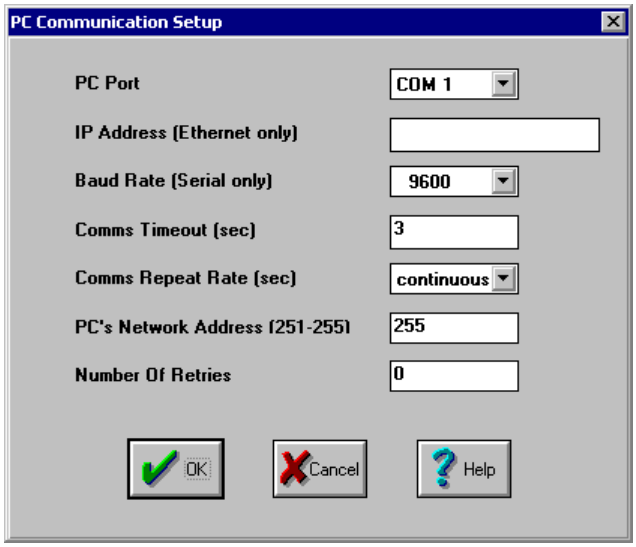
- RTU sends own pager messages: If the dial-up paging service or local GSM accepts the message or a pager radio is being used, the success counter of RTU 250 is incremented. If the dial-up paging service does not accept the message, the message is retried every 60 seconds for the maximum number of dial retries as specified for the PSTN port and then a fail is recorded. If the GSM does not accept the message (and the Dial Retries of the PSTN port is not zero), after 30 seconds the message is retried. The RTU will then retry up to 5 times in total waiting 2 minutes between retries. After all retries have failed, the fail counter for RTU250 is incremented. The fail counter of RTU 250 is also incremented when a pager message is not acknowledged (please see the Pager Message ladder block for details).
- RTU relays own pager messages: In this case, the pager message is passed on to another RTU (the pager RTU). If the pager RTU accepts the message, the success counter for RTU 250 in the local RTU will be incremented. The fail counter of RTU 250 is incremented when a pager message is not acknowledged.
- Pager RTU sends pager messages from other RTUs: this is the same as when an RTU sends its own pager messages except the RTU does not require the pager messages to be acknowledged.

Generating A Pager Message

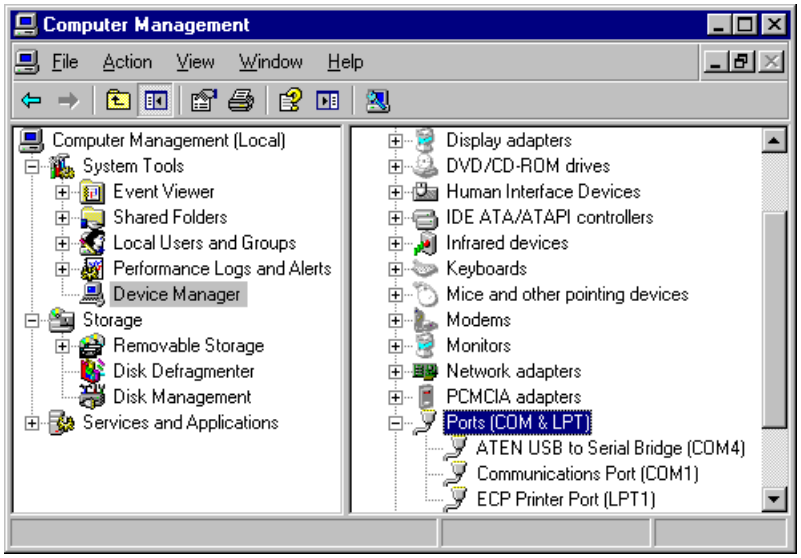
Please see the topic Example - SMS Pager Messages.

Configuration - PC Setup

Contains the Toolbox communication settings. The default settings are shown below. Usually these settings do not need to be changed except when communicating with a remote RTU over a network. When communicating over a network, it may be necessary to set Comms Timeout to 5 or more seconds and set the number of retries to 1 or more (especially when downloading an RTU configuration over the network).



PC Port: (COM 1- 48 or Ethernet) The PC's communication port to use. If using a USB to DB9 serial port converter, a COM port is assigned to the converter by the PC. To determine the converter COM port, right-click **My Computer** on the Windows Desktop and select **Manage.** Select **Device Manager** on the left hand side of the **Computer Management** window. Double-click **Ports** on the right hand side of the window. The COM port assigned to the converter will then be displayed. An ATEN converter example is shown below.



IP Address (Ethernet Only): A local area network may be used to communicate with a CP-xx RTU with an Ethernet option board. Each of the four numbers in the IP address may have values in the range of 0-255. The IP Address is only used when *PC Port* is set to *Ethernet*.

Baud Rate (Serial Only): (300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600 or 115200 bps) The baud rate used between the PC and the RTU. If PC Port is set to Ethernet, Baud Rate is ignored.

Comms Timeout (sec): The time Toolbox will wait for a response to a message attempt before performing the next message attempt or flagging a communications fail. This setting needs to be extended when communicating with an outstation RTU via a master RTU (for a 1200 baud radio network, 4 or more seconds is recommended.)

Comms Repeat Rate (sec): (Continuous, 0.1, 0.2, 0.5, 1, 2, 5, 10, 30, 60) The rate at which Toolbox generates a new message request. For example, when viewing the local registers with a Comms Repeat Rate of 0.5 seconds, a read request for local registers will be sent every 0.5 seconds. If a message fails, Toolbox will wait until the failed message has timed out (after Comms Timeout seconds) and then will send the next message at the next 0.5 second time interval. When the Comms Repeat Rate is set to Continuous or when downloading files (eg. firmware or configuration files) messages are continuously transmitted and received without pausing between messages. Note: setting the Comms Repeat Rate to Continuous may cause the PC or RTU to be overloaded with comms messages (especially if PC Port is set to Ethernet).

PC's Network Address (251-255): The PC is treated like another RTU when it sends or requests data from an RTU and so the PC must have its own unique address. Addresses 251 to 255 are reserved for this purpose. If two or more PCs are connected to the same RTU, each PC should have its own unique address to avoid communication fails. Note: SCADA software (eg. Citect) may use address 255 by default and will clash with Toolbox if Toolbox also uses address 255. Note: if the DNP3 protocol is being used in the network, PC Network Address 251 should not be used. This is because DNP3 RTU addresses above 250 use the network list entry for RTU251.

Number Of Retries: (0-9) The maximum number of attempts (after the first attempt) Toolbox will have at sending a message to the RTU if the previous attempts have failed. It is more reliable to set the Number Of Retries to 1 or more when downloading an RTU Configuration or Ladder Logic over the network.

Configuration - Download RTU Configuration

First ensure that Toolbox can communicate with the target RTU. Please see the topic **Getting Started -** Communicating With An RTU for more information.

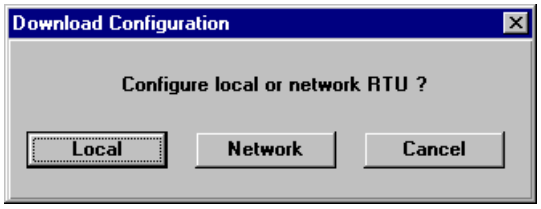
Before downloading a completely new configuration to the *local* RTU it is recommended that the RTU is first Cold Started (Utilities, Advanced, Cold Start.)

Caution! Cold starting a *remote* RTU is not recommended. After a cold start, an RTU will remember the communication settings for the first 4 ports (8 ports for a PC-1). However, it is possible that it will not be possible to communicate with the remote RTU afterwards.

Do not hot swap any module on a connected rack while downloading RTU Configuration.

Configuration, logic and ladder can be downloaded via any port, including via remote RTUs, providing a reliable connection exists.

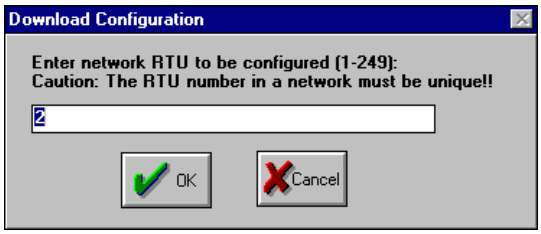
Once the local RTU has been cold started, select **Configuration, Download RTU Config**.



When attempting to download a configuration with a different address to the *local* RTU, this window will appear.

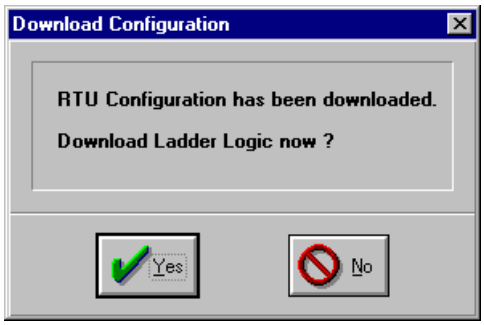
To download the configuration and update the address of the local RTU select **Local**.

To download the configuration to a remote RTU select **Network**.



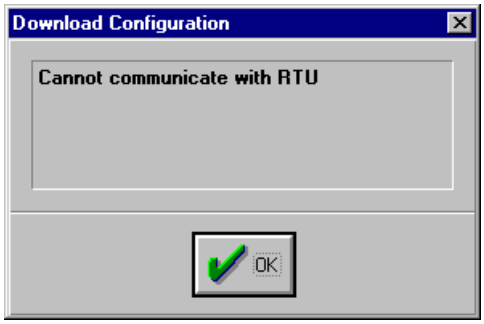
When downloading to a network (remote) RTU, enter the address of the network RTU to configure and then select **OK**.

By default, Toolbox uses the address in the configuration file.



Before ladder logic can be downloaded, it must first be compiled. Please see the topic **Ladder Logic Examples -** Downloading Ladder Logic for more information.

A warm start will occur after downloading the RTU configuration.



Please see the topic **Getting Started -** Communicating With An RTU for more information.

5. Ladder Logic

Overview	What ladder logic is and how it works
Parameters	Registers used by ladder logic blocks
Variables List	Optional: create a variables list for ladder logic. Allows data labels to be used instead of register addresses.
Editing	Editing tips. Includes how to check which registers have already been used.
Inputs	Input block details (contacts)
Outputs	Output block details (coils)
Examples	Ladder logic examples

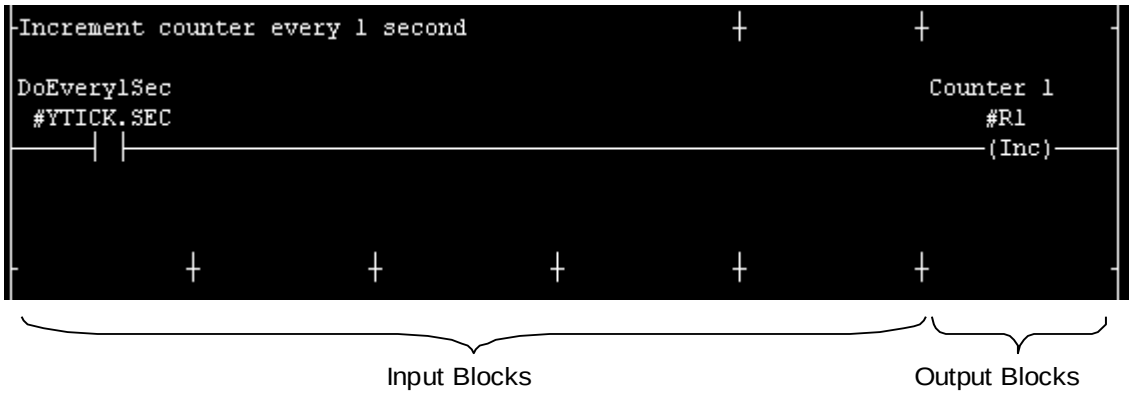
To create ladder logic, a site configuration must first be open (select **File, Open new or Existing Site**). Then select **Logic, Edit**.

Note: the default site configuration name is **NEW.SDB**. The site configuration should be saved with another name (**File, Save As**) to prevent problems when compiling ladder logic.

Microsoft Windows Vista and Windows 7 operating systems: ladder logic may not be displayed correctly. To rectify, when editing ladder logic, select **File, Select LineDraw/Text Characters**.

Ladder Logic - Overview

Ladder Logic is used to add intelligence to the RTU. It can be programmed to monitor inputs, control outputs and communicate with other RTUs or devices.



- Ladder Logic is a series of logic blocks arranged in the shape of a physical ladder
- Blocks are processed from left to right and from top to bottom
- Ladder logic blocks are configured on pages. Each page can have up to 7 lines (or rungs) of ladder logic.
- Ladder logic can be processed up to 300 times per second (depending on how busy the RTU is)
- Each ladder rung is either logically true or false
- Ladder rungs have 5 input blocks (contacts) and 1 output block (coil)
- Input blocks are used to check if something is True or False eg. is a register too high?
- Output blocks are used to do something eg. Set a value in a register or send a message
- Multiple rungs can be linked together to trigger one or more output blocks
- An output block is processed whenever a rung of input blocks are logically true and are connected to the output block
- If all paths to an output block are logically false, the output block is ignored and the next rung is processed

Multiple Ladder Files

Up to 16 ladder logic files (Filename.LL) can be added to each RTU site by using **Project, Add A Site Or Logic File** and then selecting the additional ladder file. All the ladder logic files can then be compiled into a single output file by selecting the site's SDB window and selecting **Logic, Compile**. An LLO file will be generated that has the same name as the RTU site and is now ready for downloading. Multiple ladder files are commonly used when using function blocks.

- Note 1: Please ensure that each ladder logic file is saved before compiling otherwise any unsaved changes will not be included in the compiled output file.
- Note 2: Multiple ladders are compiled in the order that they are added to the RTU site. The top ladder is compiled first. The order may be changed by editing the project file (filename.PRJ) using a text editor or by deleting the logic files and adding them again to the project.
- Note 3: Ladder logic defined after the first function block will not be regularly scanned. When adding a logic file containing function blocks to Filename.SDB, ensure that the logic file appears *under* Filename.LL in the project window.

Ladder Logic - Parameters

A Kingfisher RTU stores all I/O points, configuration settings and data variables in 16-bit *registers*. Ladder logic is able to read and write to most of these registers allowing the RTU to perform a wide range of functions - including the ability to reconfigure itself. In addition to these registers, constants and indirect addressing may also be used for ladder logic parameters (where applicable) as detailed below.

RTU Register Types

#AIss.c	Analog Input (read only)
#AOss.c	Analog Output
#DIss.cc	Digital Input (read only)
#DOss.cc	Digital Output
#Rxxxx	Local Register
#Fyyyy	Floating Point Register (32 bit)
#Lyyyy	Long Register (32 bit)
#NRrrr.xxxx	Network Register
#NArrr.ss.c	Network Analog Register
#NDrrr.ss.cc	Network Digital Register
#NFrrr.yyyy	Network Float Register
#NLrrr.yyyy	Network Long Register
#Ttt	Timer Register
#Y...	System Register
#YP...	Port Register
#YL...	Network Link Register
#YM...	Module Register

Where:
ss = Slot number, 1-64
cc = Channel number, 1-16
c = Channel number, 1-8
xxxx = Register number, 1-2048
yyyy = Register number 1,3,5…2047 (odd numbers only)
rrr = RTU address, 1-249
tt = Timer number, 1-64

A complete listing of all the available RTU registers and their descriptions is contained in the appendix - RTU Data.

Constants

Constants can be used in ladder logic in the following formats:

Integer: 0 to 65535 (default format)

Hexadecimal: 16#0 to 16#FFFF. For an explanation of hexadecimal numbers, please see Appendix - Hexadecimal Numbers.

Floating Point: Floating point numbers can have up to 7 decimal digits of precision and can have values in the range of 3.4e-38 to 3.4e+38. Floating point constants can be defined in two formats as shown below.

- [-]nnnn.nnn
- [-]nnnn.nnne[-]nnn

where *nnnn.nnn* can be up to 7 decimal digits (0-9) and [-] denotes a negative number or a negative exponential (optional). Egs. -1.0, 1.234e-3 (which can also be expressed as 0.001234). Floating point numbers must always contain a decimal point as this distinguishes them from integer constants.

Long: A signed 32 bit number in the range: -2,147,483,648 to 2,147,483,647.

Bit: 0 or 1.

Indirect Addressing

When addressing a register, the register number is usually hardcoded in the address eg. #R1. The register number can also be specified indirectly by putting the register number in another register (the pointer register) and then using indirect addressing. If the pointer register contains a value that will point out of range (eg. 0 or 3000), the indirect register address will return an incorrect value. Indirect addressing is extremely useful for reducing the amount of ladder logic required to perform a repetitious task (eg. calculating the average rainfall per minute for the last 60 minutes).

Indirect Address	Description	Example
#R[Rx] #F[Rx] #L[Rx]	Indirect local, float or long register	#R1 = 2 #R2 = 1000 #R[R1] = #R2 = 1000
#Rx.[Ra*]	Indirect local bit. Local register Ra is used to point to a bit of register Rx	#R1 = 16 #R2.16 = 1 = ON #R2.[R1] = #R2.16 = 1 (ON)
#R[Rx].[Ra*]	Indirect local register and bit	#R1 = 3 #R2 = 16 #R3 = 8000 Hex (Bit 16 ON) #R[R1].[R2] = #R3.16 = 1 (ON)
#NR[Ra*].n (n=1 to 2048)	Indirect network RTU	#R1 = 2 #NR2.1 = 1000 #NR[R1].1 = #NR2.1 = 1000
#NR[Ra*].[Rx]	Indirect network RTU and register	#R1 = 2 #R2 = 5 #NR2.5 = 1000 #NR[R1].[R2] = #NR2.5 = 1000
#NR[Ra*].[Rx].[Rb*]	Indirect network RTU, register and bit	#R1 = 2 #R2 = 1 #R3 = 16 #NR2.1.16 = 1 = ON #NR[R1].[R2].[R3] = #NR2.1.16 = 1 = ON
#R[Rx+nnn] or #R[Rx-nnn] (nnn = -128 to +127)	Indirect local register with offset. *nnn* is an offset that is added or subtracted to the pointer register Rx	Useful when a block of local registers is used to store a number of values. One register would be used as a pointer to the first register in the array, then any of the other registers in the array could be accessed from the same pointer register

* Ra, Rb and Rc must be in the range of R1 to R256 due to memory limitations of indirect addressing.
Rx can be any local register (R1 to R2048)

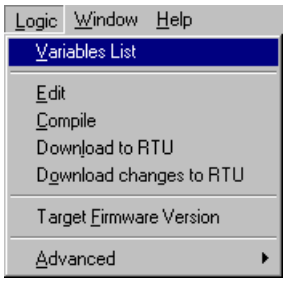
Examples
Please see the topics Example - Indirect Addressing or Example - Time Based Rolling Averages.

Ladder Logic - Variables List

Optional. Allows inputs, outputs and parameters to be defined in a list. The list can then be accessed when configuring ladder logic.

The Variables List reduces configuration errors and maintains consistency in labelling ladder logic blocks. The variables list can be used to create a database of RTU data to read and write for SCADA software.

For more than one RTU site, the variables from all the RTU sites in the project can be accessed from the variable selection list. This allows ladder blocks in one RTU site to be configured with variables from another RTU site without the need to remember register addresses or labels from that site. When using a variable from another RTU site, the variable address is automatically converted into a network address ready for use by the local RTU.



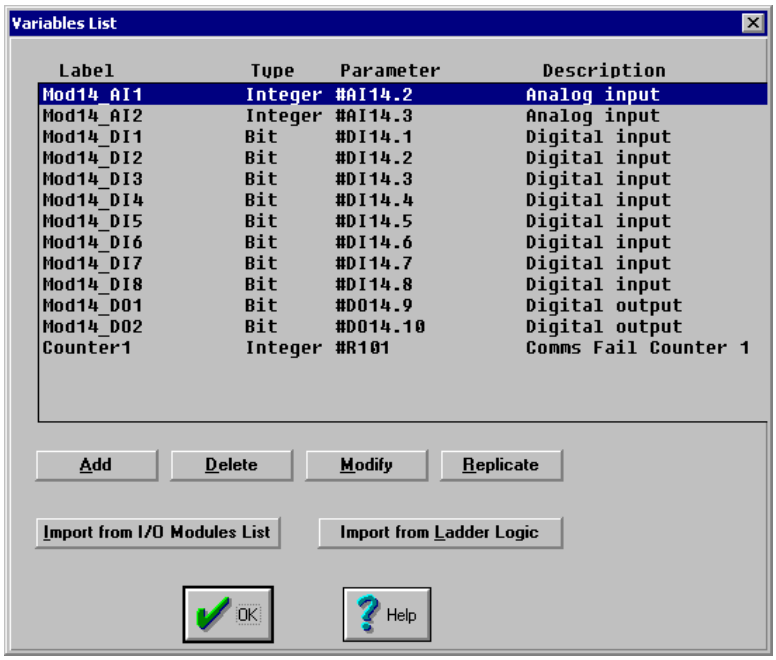
Create Variables

Open an RTU site (select **File**, **Open New or Existing Site**)

Select **Logic**, **Variables List** (this menu is only available when Filename.SDB is the active window)

Select **Add**

Enter variable parameters as detailed below



Add: Add a new variable to the variables list. Each variable has four fields as shown below.

- **Label:** (1-17 characters). Can include spaces and other ASCII characters (eg. - + < > ? !). Each label can only be used once in the variables list. Note: it is recommended that labels are 12 characters or less as only the last 12 characters are displayed in ladder logic.
- **Type:** Bit (0,1), Integer (0-65535), Long or Float.
- **Parameter:** Any Kingfisher register or constant. Each parameter can only be used once in the variables list.
- **Description:** (Optional) 0-32 character description of the variable.

Delete: Delete the selected variable.

Modify: Modify the selected variable.

Replicate: Replicate the selected variable into a number of similar variables (up to 999 variables). Please see the next section - Replicating Variables for details).

Import from I/O Modules List: Generates a variable for each data point of each module configured in the IO Modules List.

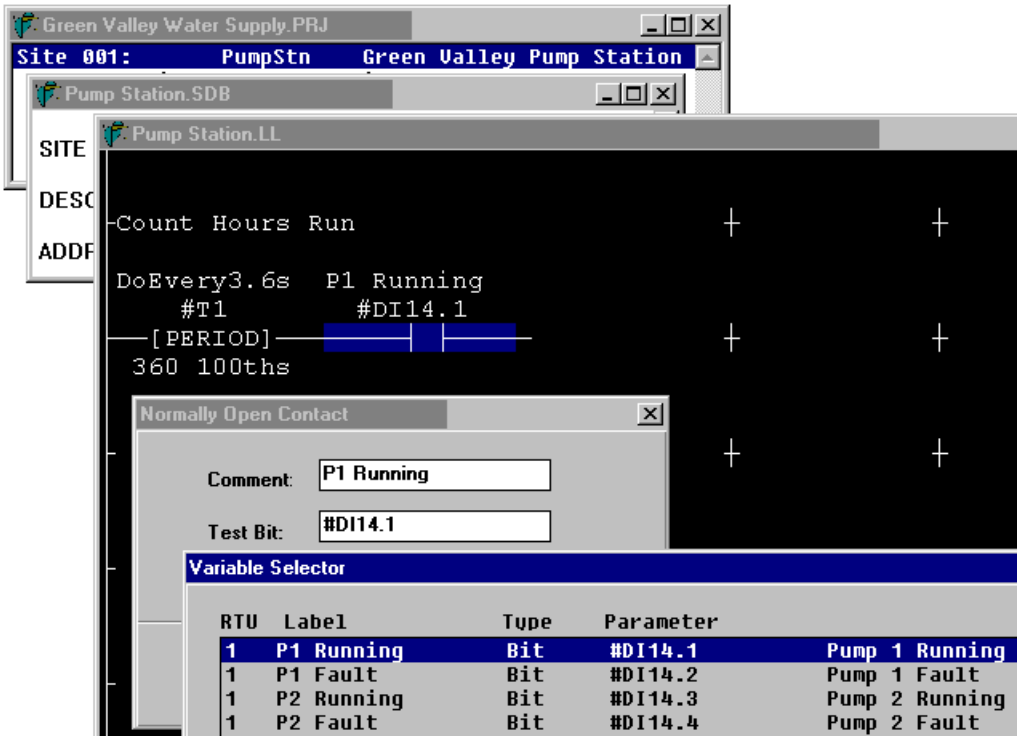
Import from Ladder Logic: Generates variables from an existing Ladder Logic file. A report file must first be generated from the ladder logic (using the menu **File, Report File** when the ladder window is displayed). This will generate a file *Filename.RPT* describing the variables used in the ladder logic. The import option then uses the RPT file to generate new entries in the Variables List. The comment for each ladder block is used as the label in the Variables List. If a comment was not used, a default label is generated.

Variables List Format
The variables list is saved as a text file *Filename.VAR* for each RTU site. A single VAR file can be used for all the RTUs in a telemetry system by creating copies of the master VAR file and renaming them to match each RTU site. The VAR file can be edited using Microsoft Notepad or Excel. Note: if using Excel, ensure the file is saved as a Text (Tab delimited) file. The file will then need to be renamed from *Filename.VAR.txt* back to *Filename.VAR*. Excel is useful for copying existing variables in the list and incrementing label names and parameters. Alternatively, the Replicate function of the Variables List can be used as detailed in the next topic.

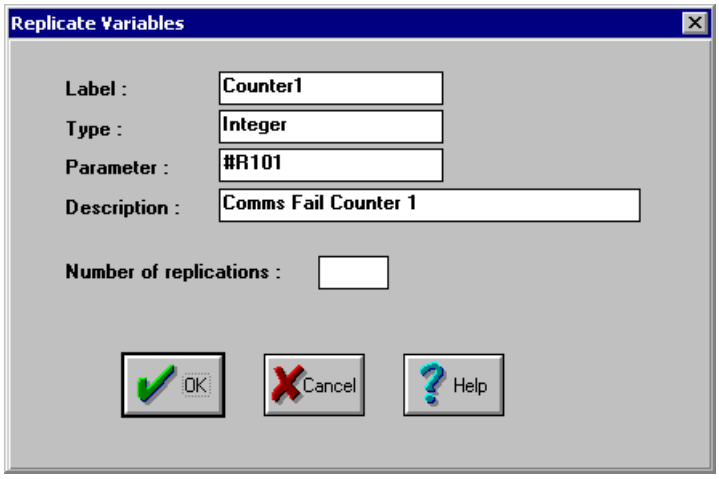
Using Variables In Ladder Logic
To access the variables list from ladder logic the RTU site must be included in a *currently open* project. Please follow the steps below.

- Open an RTU site (select **File**, **Open New or Existing Site**)
- Create some variables in the RTU site (please see above) and then close the site
- Open a new project (select **File**, **Open New or Existing Project**) and add the RTU site (select **Project**, **Add a Site or Logic File**)
- After adding the RTU site to the project, double-click on the RTU Site that is displayed in the project window
- With the RTU site open, select **Logic**, **Edit**
- Add a new ladder block or edit an existing one
- Double click on the block parameter
- The Variables List will then appear allowing a variable to be selected

Note: Block parameters can be viewed as addresses or as labels (**Ladder, View Points as Labels**). When labels are selected, Toolbox automatically searches the Variables List to find the label associated with each address. When editing ladder logic, parameters can then be entered using a label from the variables list.

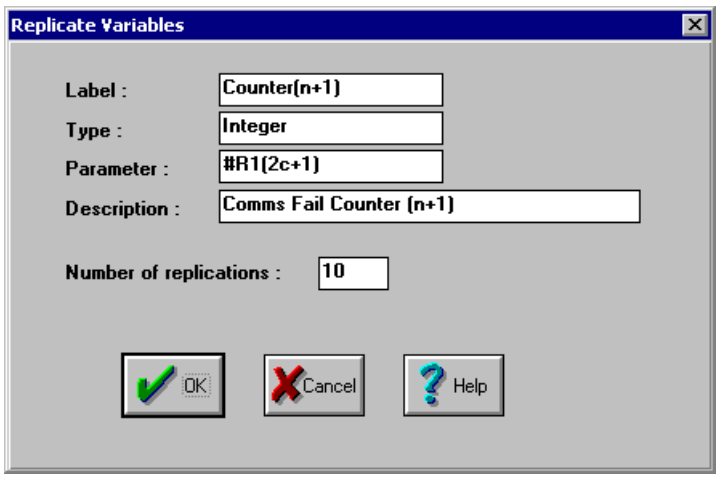


Replicating Variables

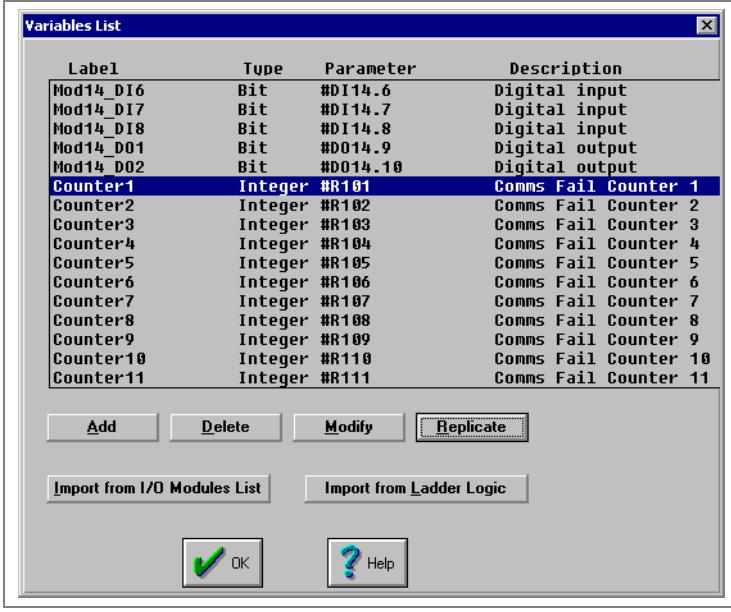


After selecting a variable in the variables list and then selecting the **Replicate** button, a replication template will appear that initially uses the settings of the selected variable.

The replication template for Counter1 is shown.



Characters in the template are replaced with wildcards or left as they are to be copied into the new variables as shown.


Variables List
Label Type Parameter Description
Mod14_DI6 Bit #DI14.6 Digital input
Mod14_DI7 Bit #DI14.7 Digital input
Mod14_DI8 Bit #DI14.8 Digital input
Mod14_DO1 Bit #DO14.9 Digital output
Mod14_DO2 Bit #DO14.10 Digital output
Counter1 Integer #R101 Comms Fail Counter 1
Counter2 Integer #R102 Comms Fail Counter 2
Counter3 Integer #R103 Comms Fail Counter 3
Counter4 Integer #R104 Comms Fail Counter 4
Counter5 Integer #R105 Comms Fail Counter 5
Counter6 Integer #R106 Comms Fail Counter 6
Counter7 Integer #R107 Comms Fail Counter 7
Counter8 Integer #R108 Comms Fail Counter 8
Counter9 Integer #R109 Comms Fail Counter 9
Counter10 Integer #R110 Comms Fail Counter 10
Counter11 Integer #R111 Comms Fail Counter 11
Add
Delete
Modify
Replicate
Import from I/O Modules List
Import from Ladder Logic
OK
Help

The variables created when **OK** is selected are shown.

The following wildcards can be used to replace characters in the replication template:

(xT+y,a,b) or **(xT-y,a,b)** or # or *****

x	(Optional) Number of characters in the starting template to modify
T	Type of conversion: *c* = character conversion, *n* = number conversion. Character conversions apply to characters in the range (A-Z, a-z, 0-9) and number conversions apply to integer numbers. A *9* will be incremented to *0* in a character conversion, but to *10* in a number conversion. The length of the text is fixed in a character conversion but it is variable in a number conversion.
+ or -	Increment (+) or decrement (-) applied before each replication
y	Amount to increment or decrement for each replication. Can also be a fraction (eg. 1/3). If a fraction, will update after 2 or more replications after a complete unit is counted.
,a,b	(Optional) Minimum limit (a) and Maximum limit (b). Only applies to number conversions.
#	Copy this character from the template variable into the new variable
*	Copy the complete string (or remaining portion of the string) from the template variable into the new variable.

Note: variables will only be replicated if each new variable has a different address and a different label to the original variable.

Example 2:

Original Label:	XYZ_Pump1Status
Replication Label:	(3c+1/2)_Pump(n+1,1,2)*
Description:	Takes the first 3 characters from the original label and increments them by one for every second new variable. Copies the next 5 characters from the original label. Takes the number in the next character position, and increments it by one for each new variable, but limited to a minimum value of 1 and a maximum value of 2. Copies the rest of the original label into each new variable.
New Variables:	XYZ_Pump2Status, XZA_Pump1Status XZA_Pump2Status XZB_Pump1Status XZB_Pump2Status

Ladder Logic - Editing

A ladder block can be edited by double-clicking on it or by pressing **Enter** when the block is highlighted. Double-clicking or pressing **Enter** on an empty ladder position will display a list of new blocks that can be added. Ladder logic can be copied, cut and pasted using the same standard key commands as a word processor as detailed below. Ladder logic can also be copied from one ladder to another if two or more sites are open in Toolbox.

Before creating ladder logic, it is useful to plan how the local registers will be used. The register plan can then be used for each RTU in the telemetry system. It can also be used to create the variables list (optional). The register plan should allow for the maximum amount of equipment that is used at all the sites (eg. pumps, valves, tanks) and should also allow for expansion. Eg. if the largest site has 4 pumps, the register map could be designed to allow for 5 pumps.

Selecting blocks: Hold the right mouse button and then drag the highlight over the required blocks. Alternatively, hold the Shift key down and then use the arrow keys.

Copying ladder blocks: Press CTRL+C or Insert. This copies the selected blocks into the memory buffer and causes the highlight to disappear. The blocks can then be pasted into ladder logic by pressing CTRL+V or Insert.

Moving ladder blocks: Press CTRL+X or delete. This copies the selected blocks into the memory buffer and deletes them from the ladder page. The blocks can then be pasted into ladder logic by pressing CTRL+V or Insert.

Changing the block type: First delete the block (by pressing Delete or CTRL+X with the highlight on the block) and then create a new block (by pressing Enter and selecting a new block type). The old block's comment and parameter settings are automatically copied into the new block.

Moving to the start or end of a rung: The Home key moves the highlight to the first block and the End key moves the highlight to the last block of the rung.

Moving to the start or end of ladder: CTRL+Home moves the highlight to the first block and CTRL+END moves the highlight to the last block of ladder logic.

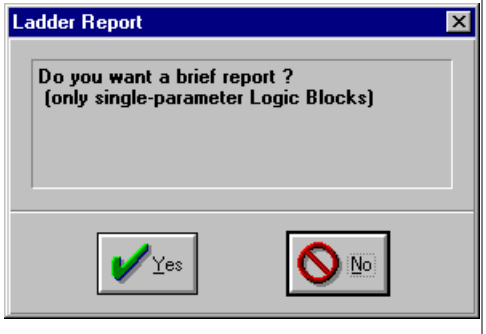
Ladder Logic Search: (CTRL+S) Searches for comment (label) or parameter in a ladder logic block. Searching is restricted to the currently displayed ladder page if the *Search This Page Only* box is ticked. Note: Search can only find direct references to registers. It cannot find the actual registers that are used by indirect addressing.

Repeat Search: (CTRL+R) Searches for the next occurrence of the block comment or parameter as entered in *Ladder Logic Search*.

Ladder Logic Translate (Search and Replace): (CTRL+T) Searches for a block comment or parameter and replaces it with the new block comment or parameter.

Register Report

Provides a list of all the register parameters used in ladder logic. Note: registers used by indirect addressing must be manually determined from the ladder logic itself.



With the Ladder Logic as the active window (in front):

Select **File, Report File**

Select **No** to obtain a comprehensive report of all the register parameters that have been used in ladder logic (including indirect register addresses). Select **Yes** to obtain a report of registers used in single-parameter blocks only (eg. contacts, coils, single copy, increment).

Filename.RPT will be created in the same folder as the RTU Configuration file. This file can be viewed using a text editor or Microsoft Excel (as a text file).

Ladder Logic - Inputs

Each type of ladder *input* block has a 12-character comment field that is used to describe the active state of the block (ie. when it is logically TRUE). For a single parameter block, the comment is used as a *tagname* (a data descriptor) to describe the active state of that parameter eg. Pump1Running. For multiple parameter blocks, the comment can be used to describe the purpose of the block eg. Lev>HiSetpt?

Input blocks include: Contact, Compare, Logical Mask, Edge Trigger and Timer.

Note: A function block Call can also be used as an input. Please see the topic Ladder Logic – Outputs, Function and Program Blocks.

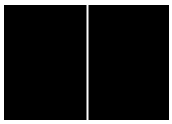
Ladder Logic - Comment Block

Can contain up to 64 ASCII characters.

Ladder Logic - Horizontal Bar

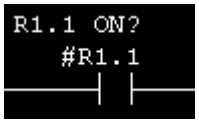
(SHIFT+F2) A horizontal bar allows ladder blocks to be connected together. A horizontal bar is always TRUE and can have a 12-character comment.

Ladder Logic - Vertical Bar


(SHIFT+F3) Each input block has a tick box *Vertical Bar*. When ticked, a vertical bar will be inserted on the right, lower side of the block. A vertical bar is always TRUE and allows blocks to be linked between rungs.

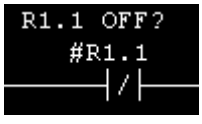
Ladder Logic - Contact

Test Bit: Any addressable bit. Eg. #R1.5, #DI14.1, #YSYS.SCAN1, #YPST2.1, #NR2.1.5, #ND2.14.1. For more Test Bit options, please see the appendix - RTU Data.



Normally Open Contact

Block is true when the Test Bit is ON (or closed). As used in the topic Example - Initialising Variables On First Scan.

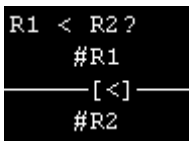


Normally Closed Contact

Block is true when the Test Bit is OFF (or open).

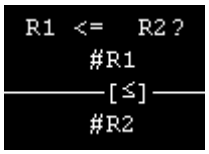
Ladder Logic - Compare

All comparisons use *unsigned* values. This means that negative numbers are treated as large positive numbers.



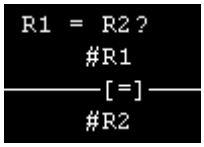
Compare: Less Than

Block is true when parameter 1 is less than parameter 2. Eg. if #R1 is less than #R2. As used in the topic Example - Exception Reporting Analogs.



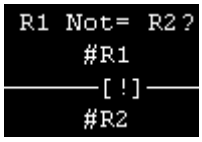
Compare: Less Or Equal

Block is true when parameter 1 is less than or equal to parameter 2. Eg. if R1 is less than or equal to R2.



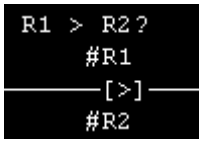
Compare: Equal To

Block is true when parameter 1 is equal to parameter 2. Eg. if R1 is equal to R2.



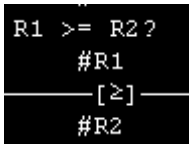
Compare: Not Equal To

Block is true when parameter 1 is not equal to parameter 2. Eg. if R1 is not equal to R2.



Compare: Greater Than

Block is true when parameter 1 is greater than parameter 2. Eg. if R1 is greater than R2. As used in the topic Example - Exception Reporting Analogs.



Compare: Greater Or Equal

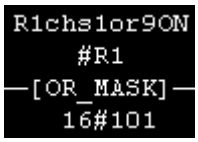
Block is true when parameter 1 is greater than or equal to parameter 2. Eg. if R1 is greater than or equal R2. As used in the topic Example - Flow Totalisation.

Ladder Logic - Logical Mask

Logical masking allows individual register bits to be selected and used for a Boolean operation. Each Logical Mask block is configured using two parameters as follows:

Test Register: Any 16-bit register.

Bit Mask: (Constant or #R) The Bit Mask parameter is normally entered as a hexadecimal number in the format 16#xxxx - where *xxxx* is the hexadecimal number. For a description of hexadecimal numbers please see Appendix - Hexadecimal Numbers. Specifying a local register allows the bit mask to be changed using ladder logic.



Logical OR Mask

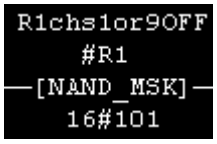
Block is true when any of the masked bits in the test register are ON. Eg. if channel 1 or channel 9 are on in #R1.

Logical AND Mask

Block is true when all of the masked bits in the test register are ON. Eg. if channels 1 and 9 are on in #R1.

Logical NOR Mask

Block is true when all of the masked bits in the test register are OFF. Eg. channels 1 and 9 are off in #R1.



Logical NAND Mask

Block is true when any of the masked bits in the test register are OFF. Eg. channels 1 or 9 are off in #R1.

Ladder Logic - Edge Trigger

Caution! After downloading ladder logic, a Positive Edge Trigger or Change block will be TRUE for one ladder scan if the test bit is TRUE. This can be prevented by also testing the system register flag #YSYS.ENABLE. Note: a warm start or a power reset will not affect these blocks.

Positive Edge Trigger

Block is true for one ladder scan when the test bit makes an OFF to ON transition (0 to 1). Eg. bit 1 of R1 makes a 0 to 1 transition. As used in the topic Example - Counting Pulses And Starts.

Negative Edge Trigger

Block is true for one ladder scan when the test bit makes an ON to OFF transition (1 to 0). Eg. bit 1 of R1 makes a 1 to 0 transition.

Change Detect

Block is true for one ladder scan when the parameter changes value. The parameter can be a single bit or a register. Eg. bit 1 of R1 makes a 0 to 1 or a 1 to 0 transition (change of state). When the parameter is a register, all 16-bits are monitored for change. As used in the topic Example - Rolling Totals Over At Midnight.

Ladder Logic - Timer

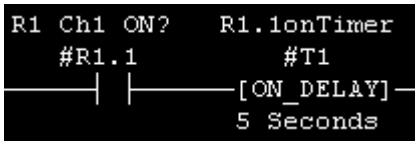
The following parameters are used by each Timer block:

Timer Register: (#T1 to #T64) Each timer register can only be used once in ladder logic. Note: for standard time intervals (eg. 1 second, 1 minute, 1 hour), it is recommended that Timer Flags (eg. #YTICK.SEC) be used instead of a Periodic Timer block. Unlike Timer registers, Timer Flags can be used multiple times in ladder logic.

Period: (Constant or #R). Specifying a local register allows the period to be changed while the ladder is running.

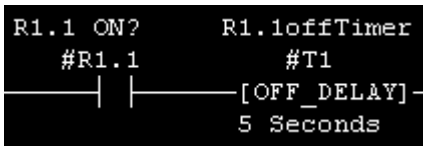
Units: 100ths (of a second), Seconds, Minutes, Hours, Days, Weeks, Months or Years.

Timer Accuracy: One Unit (as configured above). Eg a 10 minute timer will have an accuracy of 1 minute. While a 600 second timer (also 10 minutes) will have an accuracy of 1 second. Timers use the real-time clock to determine when to increment. When the specified time unit changes in the real-time clock, the timer increments. This means that if a timer is started using an ON-DELAY or an OFF-DELAY block, the timer could increment very soon or up to 1 time unit later depending on when the timer units in the real-time clock change.



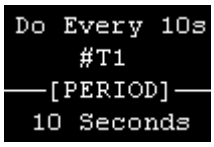
On-Delay Timer

An on-delay timer is always used with one or more contacts. The on-delay timer becomes true when the contacts on its left-hand side are true and have stayed true for at least the specified time period. When the left-hand side contacts become false, the on-delay contact becomes false also. Eg. the on-delay contact will become true when bit 1 of R1 is ON for 5 seconds and will remain true while bit 1 remains ON. As used in the topic Example - Exception Reporting Digitals.



Off-Delay Timer

An off-delay timer is always used with one or more contacts. The off-delay timer will keep the left-hand side of the rung true for the specified time period after the left-hand side of the rung becomes false. Eg. the off-delay contact will stay true (and cause the rung to remain true) for 5 seconds after bit 1 of R1 becomes false.



Periodic Timer

A periodic timer requires 3 parameters - a timer register (#Txx), the time period (0-32767) and the time units (100ths [of a second], seconds, minutes, hours, days, weeks, months, years). This contact becomes true for one ladder scan every time period. As used in the topic Example - Timer Flag. Note: if 100ths is used for the time units and the I/O Scan Interval is greater than 10 ms (ie. the ladder is scanned less than 100 times per second), the accuracy of the periodic timer will be affected.

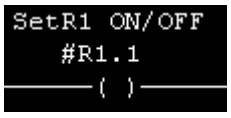
Ladder Logic - Outputs

The right-most column of ladder logic is used for Ladder Output blocks. These blocks cause something to happen - like a register bit to be set, a message to be transmitted or a calculation to be performed. Ladder Output blocks are processed when connected to a ladder rung that is logically true. The various types of ladder outputs are detailed in the following sections.

Output blocks include: Coil, Copy, Maths, Logic, Event Logging, Tx/Rx Comms, Pager Message, Function and Program Blocks, P.I.D., AGA-8, AGA-9, Clock Synchronization, Report Printer and Image Monitoring Functions.

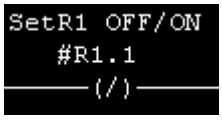
Ladder Logic - Coil

A coil parameter can be any read/write bit. These include local register bits (eg. #R1.1), hardware register bits (eg. #DO3.16), internal register bits (eg. #YDIAG.1) and network bits (eg. #NR5.1.1, #ND2.14.9).



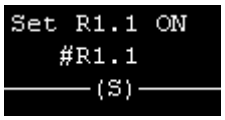
Normal Coil

When the input condition is true, the parameter is turned ON. When the input condition is false, the parameter is turned OFF. Eg. bit 1 of R1 is turned ON when the rung is true and is turned OFF when the rung is false. As used in the topic Example - Timer Flag.



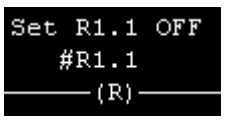
Negated Coil

When the input condition is true, the parameter is turned OFF. When the input condition is false, the parameter is turned ON. Eg. bit 1 of R1 is turned OFF when the rung is true and is turned ON when the rung is false.



Set Coil

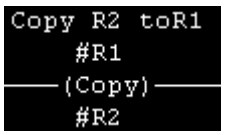
When the input condition is true, the parameter is turned ON. No action is taken when the input condition is false. Eg. bit 1 of R1 is turned ON when the rung is true and is unchanged when the rung is false.



Reset Coil

When the input condition is true, the parameter is turned OFF. No action is taken when the input condition is false. Eg. bit 1 of R1 is turned OFF when the rung is true and is unchanged when the rung is false.

Ladder Logic - Copy



Single Copy

Copies a Source bit or register (eg. #R2) to a Destination bit or register (eg. #R1).

Source: Constant, bit, 16-bit register, float or long. Any addressable bit (eg. #R1.5, #DI14.1, #YLST2.1, #NR2.1.5, #ND2.14.1) or register (eg. #R1, #F1, #DI14, #YLSUCC2, #YSEC) can be used. For more Source options, please see Appendix - RTU Data.

Destination: Bit, 16-bit register, float or long.

Type Conversions
When copying between the 3 register types (16-bit, float and long), the copy block performs a data type conversion. When a local register or a long is used as the source, it is treated as a signed number (bit 16 is the sign bit of a local register). When converting floats to 16-bit or to Long, the decimal places are truncated.

Caution! If the range of the destination register is exceeded, the result will be undefined.

Block Copy

Copies one block of registers to another block of registers or copies a constant into a block of registers.

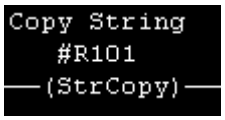
Destination: (16-bit register, float or long) The first destination register to copy to.

Source: (Constant, 16-bit register, float or long) The first source register or constant to copy from. If the source is a constant, the constant is copied into all the destination registers. If the source is a register, a block of source registers is copied to a block of destination registers. Note: it is possible to copy the contents of a single register into a block of registers by using a source register that is one less that the destination. Please see the example below.

Count: The number of consecutive registers to copy. Note: network registers are stored in blocks of 64 registers in RTU memory. When sourcing from or copying to network registers, it is not possible to cross a network register block boundary (eg. #NR1.64, #NR1.128, #NR1.192 etc). Therefore, the maximum number of network registers (#NA, #ND, or #NR) that can be block copied is 64. If the starting point is midway in a network block, then only the number of registers to the next boundary can be copied.

Examples:
Destination: #R1, Source: 100, Count: 50
Fills #R1 to #R50 with the value 100.

Destination: #R2, Source: #R1, Count: 50
Fills #R2 to #R51with contents of #R1.



String Copy

Copies up to 31 text characters (a string) to consecutive local registers. Each character is stored as an 8-bit ASCII number and the string is null terminated. Two characters are stored in each local register. For each pair of characters, the left character is stored in channels 1-8 and the right character is stored in channels 9-16. For an LP-1/2/3, the characters are stored in reverse order. Eg. string *XY* is copied to a local register

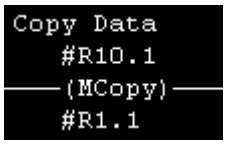
CPU Type	Chs 9-16	Chs 1-8
PC-1, CP-10/11/12/21	59 Hex (Y)	58 Hex (X)
LP-1/2/3	58 Hex (X)	59 Hex (Y)

A string copy can be used with indirect pager messages. The pager message *Line 1* is configured as a local register *#Rx*. A String Copy is then used to copy the required text to a block of registers beginning at *#Rx* before the pager message is triggered. Note: multiple strings can be joined together by copying over the null terminator of the first string with the beginning of the second string.

Destination: (#R1 - #R2048) The first local register to copy the characters to. Characters are stored in consecutive registers until the end of the string is reached (a maximum of 16 registers are used)

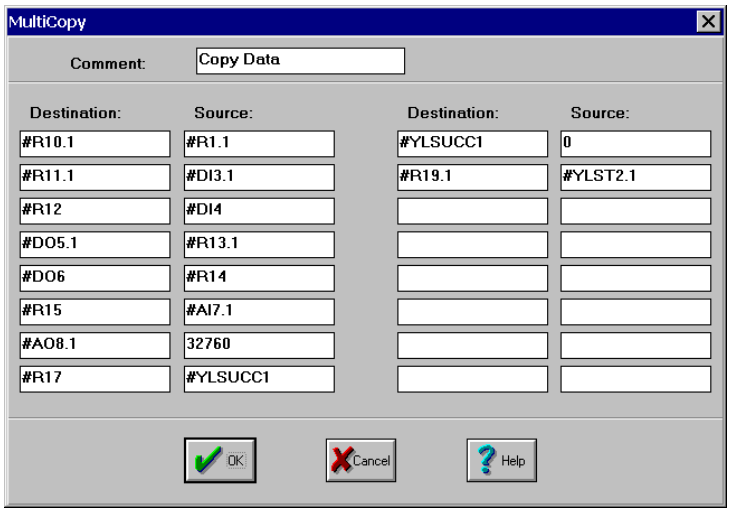
Source: Up to 31 text characters to copy to the local registers.

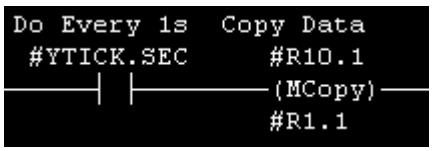
Swap Bytes? When checked, the 2 characters stored in each local register will be swapped (this function is used for the DV1000 protocol driver).



Multi Copy

A Multi Copy is the same as a Single Copy block except it can perform up to 16 individual copies at the same time. This is useful for minimising the number of ladder rungs used. A Multi Copy block showing various types of copies is shown below.



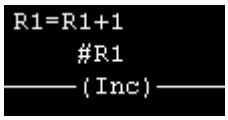


Note: multicopy blocks can use a considerable amount of processing power. To free up the processing time, multi copy blocks should be processed once a second or less by using a #YTICK.SEC contact in the multicopy rung as shown above.

Ladder Logic - Maths

16-bit Register Range
0-65535 (unsigned) or -32768 to +32767 (signed). Values overflowing the storage limit are counted from zero again. Eg. 65535 + 1 is stored as 0, 65535 + 2 is stored as 1. Values less than zero are counted backwards from 0. Eg. -1 is stored as 65535, -2 is stored as 65534. All Kingfisher addresses (ie. #R, #AI, #AO, #DI, #DO, #NR, #NA, #ND, #Y, #T) are 16-bit registers except for Floating point (#F) and Long (#L) registers which are 32-bit registers and provide a greater numerical range.

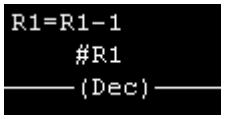
Mixing Register Types
The Add, Subtract, Multiply and Divide maths blocks allow the 3 register types (16-bit, Float or Long) to be used in any combination. If the destination register is a different type to either parameter, a type conversion is used to obtain the result.



Increment

Increments the parameter by one. Eg. R1=R1+1. As used in the topic Example - Flow Totalisation.

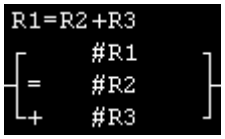
Parameter: 16-bit register (read/write) or Long register (not Float)



Decrement

Decrements the parameter by one. Eg. R1=R1-1.

Parameter: 16-bit register (read/write) or Long register (not Float).



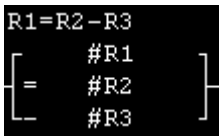
Add

Parameter 2 (R3) is added to Parameter 1 (R2) and the result is put in the Destination (R1). As used in the topic Example - Flow Totalisation.

Destination: 16-bit register (read/write), Long or Float register.

Parameter 1, Parameter 2: 16-bit register (read/write), Long, Float or constant. Register types can be mixed in any order.

Caution! It is possible to exceed the range of the destination register and produce an undefined result.



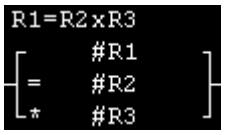
Subtract

Parameter 2 (R3) is subtracted from Parameter 1 (R2) and the result is put in the Destination (R1). As used in the topic Example - Flow Totalisation.

Destination: 16-bit register (read/write), Long or Float register.

Parameter 1, Parameter 2: 16-bit register (read/write), Long or Float register or constant. Register types can be mixed in any order.

Caution! It is possible to exceed the range of the destination register and produce an undefined result.



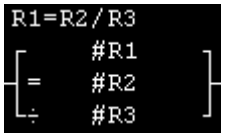
Multiply

Parameter 2 (R3) is multiplied with Parameter 1 (R2) and the result is put in the Destination (R1). The Multiply block treats 16-bit registers as signed numbers (-32767 to +32767, highest bit = sign).

Destination: 16-bit register (read/write, signed), Long or Float register.

Parameter 1, Parameter 2: 16-bit register (read/write, signed), Long or Float register or constant. Register types can be mixed in any order.

Caution! It is possible to exceed the range of the destination register and produce an undefined result.



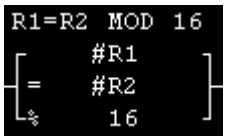
Divide

Parameter 1 (R2) is divided with Parameter 2 (R3) and the result is put in the Destination (R1). The Divide block treats 16-bit registers as signed numbers (-32767 to +32767, highest bit = sign).

Destination: 16-bit register (read/write, signed), Long or Float register.

Parameter 1, Parameter 2: 16-bit register (read/write, signed), Long or Float register or constant. Register types can be mixed in any order.

Caution! It is possible to exceed the range of the destination register and produce an undefined result. Note: the result is undefined after a divide by zero.



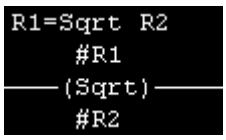
Modulus

Calculates the modulus of Parameter 1 divided by Parameter 2 and returns the result in the Destination. The modulus is the remainder after division and is represented by the percentage symbol (%). Eg. 10 % 3 = 1 (10 divided by 3 equals 3, with a remainder of 1). The Modulus block treats 16-bit registers as signed numbers (-32767 to +32767, highest bit = sign).

Destination: 16-bit register (read/write, signed) or Long register (not Float).

Parameter 1, Parameter 2: 16-bit register (read/write, signed), Long register or constant (not Float). Register types can be mixed in any order.

Caution! It is possible to exceed the range of the destination register and produce an undefined result.

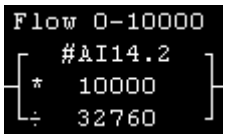


Square Root

The square root of the Source (R2) is put in the Destination (R1).

Destination: Float register (#F).

Source: Float register (#F) or constant.



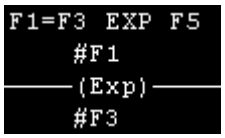
Multiply/Divide

Multiplies the *Source* with the *Multiply By* parameter, then divides the result with the *Divide By* parameter and then puts the result in *Destination*. The Multiply/Divide block treats 16-bit registers as signed numbers (-32767 to +32767, highest bit = sign).

Destination: 16-bit register (read/write, signed).

Source, Multiply By, Divide By: 16-bit register (read/write, signed) or constant. Note: a divide by zero causes the destination to remain unchanged.

The Multiply/Divide block is very useful for scaling analog values into engineering units within the RTU. The above example shows an analog input being converted to a number in the range 0-10,000 which could then be displayed as 0-100.00% (Series 2 analog inputs are stored as a number in the range 0-32760 = 0-100%). The Multiply/Divide block allows high accuracy when scaling a number as it uses a 32 bit total for its calculations and then returns the lowest 16-bits as the result. If the result is greater than 65535 (16-bit limit), the Multiply/Divide block returns a value of 8000 Hex (32767). Note: It is generally recommended to use separate Multiply and Divide blocks with floating point parameters for scaling calculations as they can store a much wider range of numbers, including fractional numbers, and are not likely to go out of range.

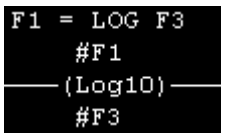


Exponential

Calculates Parameter 1 (F3) to the power of Parameter 2 (F5) and returns the result in the Destination (F1).

Destination: Float register.

Parameter 1, 2: Float register or constant.

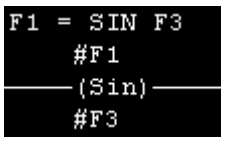


Logarithm

Calculates the Logarithm (base 10) of the Source (F3) and stores the result in the Destination (F1).

Destination: Float register.

Source: Float register or constant.

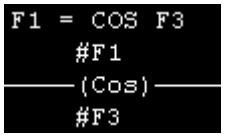


Sine

Calculates the Sine of the Source (F3) and stores the result in the Destination (F1). Angles are defined in radians. Note: $1^{o} = 2\pi/360 = 0.017453$ Radians.

Destination: Float register.

Source: Float register or constant.

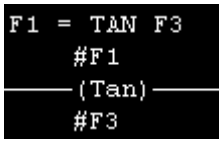


Cosine

Calculates the Cosine of the Source (F3) and stores the result in the Destination (F1). Angles are defined in radians. Note: $1^{o} = 2\pi/360 = 0.017453$ Radians.

Destination: Float register.

Source: Float register or constant.

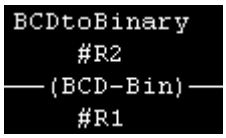


Tangent

Calculates the Tangent of the Source (F3) and stores the result in the Destination (F1). Angles are defined in radians. Note: $1^{o} = 2\pi/360 = 0.017453$ Radians.

Destination: Float register.

Source: Float register or constant.

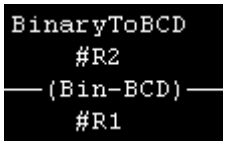


BCD To Binary

Converts a BCD number (#R1) to binary format (#R2). BCD uses a group of 4 bits to represent each decimal digit (0-9). Not supported by CP-21.

Destination: 16-bit register (read/write).

Source: 16-bit register (read/write) or constant.



Binary To BCD

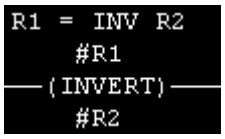
Converts a binary number (#R1) to BCD format (#R2). BCD uses a group of 4 bits to represent each decimal digit (0-9). Not supported by CP-21.

Destination: 16-bit register (read/write).

Source: 16-bit register (read/write) or constant.

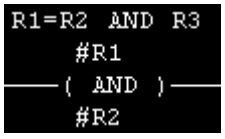
Ladder Logic - Logic

For each of the following Logic Functions, the input parameters can be 16-bit registers or constants. These functions all perform bit-wise operations; ie. they treat the registers as 16 individual bits.



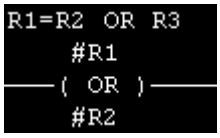
Invert

The Source (R2) is inverted, and placed in the Destination (R1).
An Invert causes all the 16-bits in the register to be changed (ie. 1's are changed to 0's and 0's are changed to 1's).



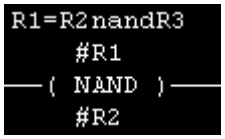
AND

Parameters 1 and 2 (R2, R3) are ANDed together and the result is placed in the Destination (R1). This means only the bits which are a *1* in *both* parameters, will be a *1* in the destination. All other bits will be 0.



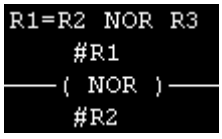
OR

Parameters 1 and 2 (R2, R3) are ORed together and the result is placed in the Destination (R1). This means that all the bits which are a *1* in *either* parameter will be a *1* in the destination. All other bits will be 0.



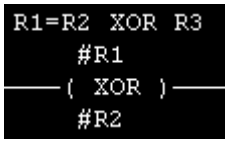
NAND

Parameters 1 and 2 (R2, R3) are NANDed together and the result is placed in the Destination (R1). This means that the bits which are a *1* in *both* parameters will be a *0* in the destination. All other bits will be 1.



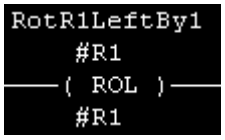
NOR

Parameters 1 and 2 (R2, R3) are NORed together and the result is placed in the Destination (R1). This means that all the bits which are a *1* in *either* parameter will be a *0* in the destination. All other bits will be 1.



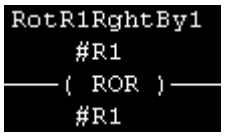
XOR

Parameters 1 and 2 (R2, R3) are XORed together and the result is placed in the Destination (R1). This means that all the bits which are the same in *both* parameters will be a *0* in the destination. All other bits will be 1.



ROL (Rotate Left)

Parameter 1 is rotated left by the number of bits specified in Parameter 2, and the result placed in the Destination.
Eg. Destination = #R1, Parameter 1 = #R1, Parameter 2 = 1
If #R1 initially contains the value 10 hex (0000 0000 0001 0000 binary), it will contain 20 hex (0000 0000 0010 0000 binary) after calling this function. If #R1 initially contains the value 8000 hex (1000 0000 0000 0000 binary), it will contain 1 hex (0000 0000 0000 0001 binary) after calling this function.



ROR (Rotate Right)

Parameter 1 is rotated right by the number of bits specified in Parameter 2, and the result placed in the Destination.
Eg. Destination = #R1, Parameter 1 = #R1, Parameter 2 = 1
If #R1 initially contains the value 2 hex (0000 0000 0000 0010 binary), it will contain 1 hex (0000 0000 0000 0001 binary) after calling this function. If #R1 initially contains the value 1 hex (0000 0000 0000 0001 binary), it will contain 8000 hex (1000 0000 0000 0000 binary) after calling this function.

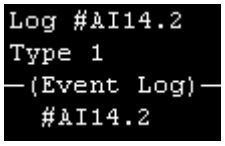
Ladder Logic - Event Logging

Event logs allow the RTU to record time and date stamped data. An event log can be created periodically, after data changes or on any configurable event.

Event logs are kept in a circular buffer that is "max. number of logs" long (as detailed by the Memory configuration, **Check memory usage** button). When the buffer is full, the oldest logs are overwritten. The RTU uses an internal "current pointer" which always points to the latest log added to the buffer.

Accumulating Event Logs From Other RTUs
Event logs received from other RTUs are all stored in one event log buffer. To keep track of which logs have been received, a master RTU uses a local log pointer for each outstation RTU. Each of these pointers is then used to point to the last log polled from the outstation RTU. For greater flexibility, a local log pointer can be set to point to the latest log in an outstation RTU or it can be set to point to a log that occurred a number of minutes ago in the outstation RTU.

RTU Communications
Most of the event log blocks initiate messages and should be used in a similar way to other communication blocks by first checking if the port is available before sending a message. For an example of using event log blocks, please see the topic Example - Polling Event Logs.



Event Log

Logs the value or state of a variable along with the user type and priority of the event log. Please see the topic Example - Event Logging. Note: a maximum of 250 Event Log blocks can be used per RTU.

Comment: A 12-character description.

Ref: No longer used.

Variable: Bit, 16-bit register, Long (#L) or Float (#F).

User Type: (0-31) Used to group similar types of logs. For example, analog inputs could be type 1, digital status signals type 2, digital alarm signals type 3 and so on. Only logs matching a certain User Type can then be uploaded instead of uploading all the logs.

Priority: (0-7) Allows separation of logs within each User Type category. 0 is used for the highest priority logs.

Tx Event Logs

No longer supported. Please use Tx Update Event Logs block.

Tx Update Event Logs

This block is designed to update event logs in a standby master RTU. It also updates the event log pointers for the remote RTUs in the standby master. The block checks if new logs need to be transmitted to the destination RTU and then sends them 10 at a time until it has sent the maximum limit of event logs or until the end of the event log list is reached. Requires driver *TXUPDATE.Dxx*. Care must be taken to initiate only one Tx or Rx Update block at a time otherwise unpredictable results may occur. The pending flag detailed below can be used to determine when the Tx Update block has finished.

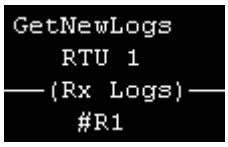
Destination RTU: (1-249) The target RTU to send event logs to.

Event Log Pointer: (#R) The local register which the RTU uses to remember where it is up to in the event log list. It is automatically updated after the TX Update block is successfully completed.

Status Register: (#R) A local register used to indicate the status of the block as follows:

- Channel 1: Pending Flag. Channel 1 is set ON when the block is activated and set OFF when the block is finished.
- Channel 2: Status Flag. Channel 2 is written to after the block is finished. Channel 2 is set OFF if the update was successful or is set ON if the update failed (due to communications failure).
- Channel 3: Finished Flag. Channel 3 indicates whether the Event Log list contains any more entries and is written to after a block of event logs has been successfully transferred. Channel 3 is set ON if all the event logs have been sent or is set OFF if there are more logs.

Max. number of logs: (0-65535) The maximum number of logs to transmit each time the Tx Update Event Logs block is activated.



Rx Event Logs

No longer supported. Please use the Rx Update RTU Info block.

Rx Event Logs from Specific Period

Polls event logs that occurred over a specific time period from a remote RTU. It will keep polling groups of 10 logs at a time until it has received the maximum limit of logs or until the end of the event log list is reached.

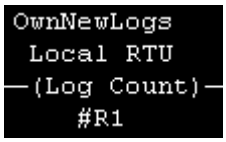
RTU: (1-249) The target RTU to poll the event logs from.

Status Register: (#R or blank) When a local register is specified, the channels are defined as follows:

- Channel 1: Pending Flag. This bit is set ON when the block is activated and then set OFF after the block has finished.
- Channel 2: Status Flag. This bit is written after the block is finished. The status flag is set OFF if the block was successful or set ON if the block failed (communications failure).

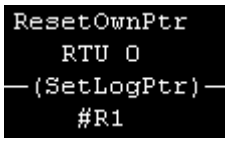
Start Time (minutes before now), Period: A constant (0-32767) or a local register (#R). These fields are used to specify the time period for uploading logs.

Filter Logs by: Only event logs that match the specified priority or user type are uploaded. The maximum number of logs to upload can also be specified.



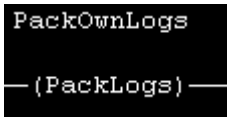
Get Event Log Count

No longer supported.



Set Event Log Pointer

No longer supported.



Pack Event Logs

Compacts the local Event Log list by deleting event logs that are older than the specified period. Not supported by CP-21.

Log Retention Period: These fields indicate time in hours before now. They specify the period for which logs of each priority will be retained. All event logs older than the specified period will be deleted.

Warnings:

- The *Pack Event Logs* block will cause event logs to be re-ordered in the event log buffer. Therefore any pointers to the event log buffer which are not current will be invalid and using these pointers for transferring logs will result in inconsistent behaviour (logs may be uploaded twice or out of order).
- It is possible to generate many thousands of event logs in an RTU. Once the event log list becomes large, any blocks which require searching through this list will become slow, and will therefore cause delays in scanning I/O points and processing ladder logic.

Ladder Logic - Tx / Rx Comms

The following *Kingfisher* blocks are detailed below: Tx/Rx Data, Tx/Rx Images and Rx Update.

Please see the Communication Drivers chapter for more protocol drivers.

Ladder Logic - Transmit / Receive Data

Whenever data is transferred from one RTU to another, the data is always placed in network registers. When communicating with an RTU connected with a PSTN modem link, the RTU will automatically dial the number configured for that RTU.

Series 2: Transmit Data

Transmits up to 32 16-bit registers to a destination RTU. As used in the topic Example - Sending The Exception Report.

Comment: A 12-character description.

RTU #: (1-249) The destination RTU that the data is sent to (note: addresses 250-255 are reserved for paging and PC use).

Registers: The registers to transmit to the destination RTU. Can enter #R, #F, #L, #DI, #AI, #NR, #ND or #NA registers in any order. Note: If one or more of the registers is a network register (#N…), then a maximum of 25 registers can be entered. Float (#F) and Long (#L) registers count as two registers each. The 16 channels of a digital module can be transferred as one 16-bit register (eg. #DI14) while analog channels must be transferred individually (eg. #AI15.1, #AI15.2) as each channel is stored in one register. For more details about each type of register, please see the appendix - RTU Data.

Series 2: Receive Data

Polls up to 32 registers from a source RTU. As used in the topic Example - Polling Data.

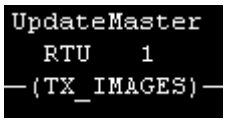
Comment: A 12-character description.

RTU #: (1-249) The source RTU that the data is received from (note: addresses 250-255 are reserved for paging and PC use).

Registers: The registers to poll from the source RTU. Can enter #R, #F, #L, #DI, #AI, #NR, #ND or #NA registers in any order. Note: If one or more of the registers is a network register (#N…), then a maximum of 25 registers can be entered. Float (#F) and Long (#L) registers count as two registers each. For more details about each type of register, please see the appendix - RTU Data.

Ladder Logic - Tx/Rx Update Network RTU Images

Care must be taken to initiate only one Tx or Rx Update block at a time otherwise unpredictable results may occur. The pending flags detailed below can be used to determine when the block has finished before generating new Tx or Rx update messages.



Series 2: Tx Update Network RTU Images

Sends new network data to a destination RTU. It is possible to update the network data for up to 32 RTUs using one TX Images block. Network data is stored in an RTU as blocks of 64 registers. The TX Images block works by requesting the CRC for each block from the destination RTU (*RTU #*) and if the CRCs are different, the local RTU updates the block in the destination RTU. Only network blocks that are different are updated which minimises communication time. Requires driver *TXUPDATE.Dxx*. As used in the appendix topic Redundancy, Redundant RTUs.

Comment: A 12-character description.

RTU #: (1-249) The destination RTU that the network data is sent to.

RTUs: (1-249) The network data for these RTUs is compared between the destination RTU and the local RTU and then the data in the destination RTU is updated if necessary.

Series 2: Rx Update Network RTU Images

The RX Images block is used to get new network data from a source RTU. It is possible to update the network data blocks of up to 16 RTUs at a time using one RX Images block. An RX Images block works by requesting the CRCs for each network block from the source RTU. If the CRCs are different to the local RTU, the local RTU polls the new network blocks from the source RTU. Only network blocks that are different are updated which minimises communication time. Requires driver *RXUPDxx.Dxx*.

Comment: A 12-character description.

RTU: (1-249) The source RTU to receive network data from.

RTUs to update: (1-249) The network data blocks (images) of these RTUs are requested from the source RTU. The network data for up to 16 RTUs can be requested at once.

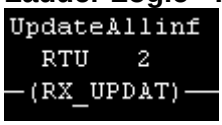
Selection and Status Controls
These fields control the RTU images to update and indicate the current status of the block.

Realtime Data Mask: Local register (#R), *ALL* (all RTUs in the list are updated), or *NONE* (none of the RTUs in the list are updated). When a local register is used, the 16 register channels correspond to the 16 RTUs to update respectively. When a channel is set ON, the data for the corresponding RTU will be updated. After a successful data update, the channels are not reset. The default value for this field is *ALL*.

Pending flags: *NONE* or a local register (#R). Each of the 16 channels indicates the pending status of the corresponding RTU in the *RTUs to update* list. All the channels are set ON when the RX Images block is activated. After the network data for each RTU is updated, the channel corresponding to that RTU is set OFF.

Status flags: *NONE* or a local register (#R). Each of the 16 channels indicates the success/failure status of the corresponding RTU in the *RTUs to update* list. Each flag is written to after the network data of that particular RTU has been updated. A flag is set OFF if the update is completed successfully or is set ON if the update has failed.

Ladder Logic - Rx Update RTU Info



Series 2: Rx Update RTU Info

Polls data and event logs from up to 16 RTUs. An RX Update block can also issue a *Sync Clock* command to each of the RTUs. The Rx Update block works by requesting the CRC for each network block from each RTU and then requests the blocks that have changed. Only network blocks that are different are updated which minimises communication time. Event logs that match the priority and user type are uploaded until the maximum limit is reached or until there are no more event logs. Requires driver *RXUPDxx.Dxx*. Care must be taken to initiate only one Tx or Rx Update block at a time otherwise unpredictable results may occur. The pending flags detailed below can be used to determine when the Rx Update block has finished. As used in the topic Example - Polling Event Logs.

Note: for local and hardware registers, the source RTU controls which of its network data blocks will be checked or uploaded. The system parameters *Update Register Blocks* and *Update Hardware Blocks* are configured in the outstation RTU to control this function (please see the topic Configuration - System Parameters).

Comment: A 12-character description.

RTUs to update: (1-249) A list of up to 16 RTUs to request data from.

Selection Controls
These fields identify which update functions (update real-time data, update event logs, synchronize clocks) apply to each of the listed RTUs. If a local register (#R) is specified, the 16 channels correspond to each of the 16 RTUs to update. When a channel is set ON, the corresponding RTU will be updated. Alternatively, *ALL* (all RTUs in the list are updated), or *NONE* (none of the RTUs in the list are updated) can be specified. The default value for each of these fields is *ALL*.

- **Realtime Data mask:** If a register channel is set ON, real-time data will be polled from the corresponding RTU. The channel is NOT reset after a successful data update.
- **Event Logs mask:** If a register channel is set ON, event logs will be polled from the corresponding RTU according to the *Event Log Control* fields. The channel is reset when all event logs have been received from the RTU.
- **Sync Clocks mask:** If a register channel is set ON, the clock of the specified RTU will be synchronized to the local RTU's own clock. The channel is reset if the RTU is synchronized successfully.

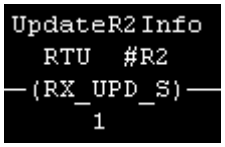
Status Controls
These fields indicate the current status of the Rx Update function. When set to *NONE*, the status controls are not used.

Pending flags: *NONE* or a local register (#R). Each of the 16 channels indicates the pending status of the corresponding RTU in the *RTUs to update* list. Each channel is set ON when the RX Update block is activated and then set OFF when polling of that particular RTU has finished. Note: the local register is not automatically set to zero after a warm start.

Status flags: *NONE* or a local register (#R). Each of the 16 channels indicates the success/failure status of the corresponding RTU in the *RTUs to update* list. Each flag is written to after polling of the particular RTU has finished. A flag is set OFF if the update is completed successfully or is set ON if the update has failed.

Event Log Controls
Max. Logs to Upload: (0-32760) The maximum number of logs to upload each time the RX Update block is activated.

Priority, User Type: Only event logs that match the Priority and User Type settings are retrieved.



Series 2: Rx Update RTU Info, single RTU

Polls data and event logs from a single RTU. An RX Update block can also issue a *Sync Clock* command to the RTU. The Rx Update block works by requesting the CRC for each block and then requesting the blocks which have changed. Only network blocks that are different are updated which minimises communication time. Event logs that match the priority and user type are uploaded until the maximum limit is reached or until there are no more event logs. Requires driver *RXUPDxx.Dxx*. Care must be taken to initiate only one Tx or Rx Update block at a time otherwise unpredictable results may occur. The pending flag detailed below can be used to determine when the Rx Update block has finished.

This block is also useful for copying event logs from a submaster RTU to a master RTU. The Rx Update Single block ignores the source of the event logs in the submaster RTU and simply copies all the logs from the submaster RTU to the master RTU.

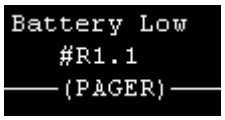
Note: for local and hardware registers, the outstation RTU controls which of its network data blocks will be checked or uploaded. The system parameters *Update Register Blocks* and *Update Hardware Blocks* are configured in the outstation RTU to control this function (please see the topic Configuration - System Parameters).

Fields are the same as for an Rx Update block for multiple RTUs except for:

Control Register: Local register (#R) or blank. A blank entry causes real-time data and event logs to be updated, the clock to be synchronized and to not use the pending or status flags. When a local register is specified, the channels must be configured as follows:

- Ch1: Real-time Data Flag. If this channel is set ON, real-time data will be polled from the specified RTU. Channel 1 is NOT reset after a successful data update.
- Ch2: Event Logs Flag. If this channel is set ON, event logs will be polled from the specified RTU according to the *Event Log Control* fields. Channel 2 is reset when all event logs have been retrieved from the RTU.
- Ch3: Sync Clock Flag. If this channel is set ON, the clock of the specified RTU will be synchronized to the local RTU's own clock. Channel 3 is reset if the RTU is synchronized successfully.
- Ch4: Pending Flag (set by Rx Update block). Indicates the pending status of the RX Update block. Channel 4 is set ON when the block is activated and is set OFF when the block has finished.
- Ch5: Status Flag (set by Rx Update block). Indicates the success/failure status of the RX Update. Channel 5 is written to after polling of the RTU has finished. Channel 5 is set OFF if the update is completed successfully or is set ON if the update has failed.

Ladder Logic - Pager Message



Pager Message

Sends a 32-character pager message to up to 12 pager receivers. The pager message can also be sent with a time and date stamp. Requires driver *PAGINGxx.Dxx*.

A pager message will initially be sent to the pager receiver(s) configured in Group 1 of the selected pager sequence. If an acknowledge is received within the time specified in the **Wait for Ack** field (by writing a 0 to the acknowledge bit), the sequence is completed and no further action is taken. If an acknowledge is not received, the same message is then sent to the Group 2 pagers (if any pagers have been configured for Group 2). If an acknowledge is not received within the Group 2 **Wait for Ack** time, the message is sent to the Group 3 pagers. If an acknowledge is not received within the Group 3 **Wait for Ack** time, the RTU will flag a fail for RTU250 (RTU250 is reserved for paging statistics) and increment the fail counter. Please see the topic Example - SMS Pager Messages.

Comment: A 12-character description.

Message: A local register (#R1 - #R2048 specified on *Line 1*) *or* up to 2 lines of 16 characters. A local register can be used to point to the beginning of a block of registers that contain the pager message text. When using a local register, no other text can be included on *Line 1* or *Line 2*. The String Copy block is used to store text characters in local registers for use by the pager message block.

Prepend Site Address & Name: If ticked, *RTU Addr: xxx Site: SITENAME* is added to the front of the message; where *xxx* is the RTU address and *SITENAME* is the 8-character Site Name of the SDB file loaded in the RTU.

Append Date/Time Stamp: If ticked, a time and date stamp is added to the end of the pager message as follows *DD-MM HH.MM.SS* - ie. day-month hour.minutes.seconds. The year is not included.

Pager Seq (1-5): Indicates which pager sequence to use for this message (as configured in **Configuration, Pager Configuration**). A pager sequence defines which pager receivers to send the pager message to.

Acknowledge Bit: This bit is set when the pager message is first sent. Resetting the bit to zero (eg. using SCADA software) will acknowledge that the pager message is received and no further messages will be sent. If this bit is not cleared, a fail will be recorded for RTU250 once the pager message has been sent to all the configured pager numbers and the last **Wait for Ack** time has expired (RTU250 is reserved for paging statistics).

For more information, please see the topics Configuration - Pager Configuration and Example - SMS Pager Messages.

Ladder Logic - Function And Program Blocks

Function Blocks allow commonly used pieces of logic to be defined once and then re-used as many times as necessary. Up to 32 variables can be passed to a function block and then used within the function block. Variables are referenced by "%number"; eg. the third parameter would be referenced as "%3".

A function block definition commences with a *Start Function Block*, and ends with a *Return from Function Block*. The function block is called or run from ladder logic by using a *Call Function Block*.

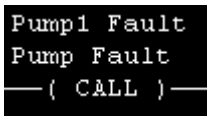
A function block can be configured as an input by defining a Boolean (true or false) return parameter, or as an output by not defining a return parameter as detailed below. A function block can also be used to call another function block. This can occur multiple times so that one function block call can trigger a string of function block calls.

Rules for Configuring Function Blocks

- All function blocks must be placed at the end of a ladder logic configuration after the normal (main loop) ladder logic. The compiler will search for the first *Function Block Start* and terminate the normal ladder logic there.
- All function blocks must end with an unconditional *Return from Function Block*, ie. one directly connected to the left power rail. However, a function block may contain any number of conditional returns.
- Function block names are case sensitive: the name in a *Call Function Block* must exactly match the name in a *Start Function Block*.
- The number of variables and the data types in a *Call Function Block* must exactly match those defined in the corresponding *Start Function Block*.
- A function block defined without any return parameter can only be called from an output position (the right-most column).
- A function block defined with a Boolean return parameter can only be called from an input position (any column except the right-most one).
- Timer and edge trigger ladder blocks cannot be used within function blocks. (Note: a timer can be created using #YTICK.SEC to increment or decrement a register. An edge trigger requires the current value to be compared to the previous value (a bit or register), to see if it has changed. The current value is then copied to the previous value for use in the next scan.)
- Up to 500 Call, Jump and Start Function blocks (in total) can be used per RTU.

Function blocks defined in another ladder logic file may be called by the main ladder logic file by using a Project file. This is useful when there are many RTU sites that all use the same piece of ladder logic. If a change needs to be made, only the one ladder logic file needs to be updated and then the ladder logic for each site is re-compiled. Please see the topic Ladder Logic, Multiple Ladder Files.

Function blocks for DNP3 objects are available from http://helpdesk.servelec-semaphore.com.



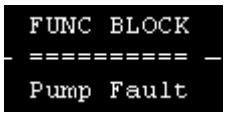
Call

Calls a function defined in the ladder. As used in the topic Example – Time Based Rolling Averages.

Call Function: The name of the function block to call. This name must be entered exactly as it appears in the corresponding Start Function block (it is case sensitive). To be used as an input, the function block called must have a return variable of *Boolean*. To be used as an output, the function block called must have a return variable of *None*.

Variables: A list of all variables (registers, constants, etc) to be passed to the function block. Eg. the above call function passes pump 1 parameters to the *Pump Fault* function. *Pump Fault* then returns True or False. Note: Ladder logic processing is transferred to the specified function block, until a *Return from Function Block* is encountered. Ladder then continues being processed from after the function block call.

Note: DNP3 function blocks are available from http://helpdesk.servelec-semaphore.com.



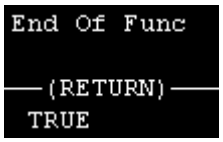
Start Function Block

Defines the start of a function block. It must be located on its own ladder rung above the function block ladder logic that it denotes the start of. As used in the topic Example – Time Based Rolling Averages.

Function Name: The name of the function block. This is a 12-character case-sensitive name that can include spaces.

Variables: (Optional) A function block can be passed up to 32 variables when it is called using the Call function block. The data type of each variable passed must be defined here as Boolean (bit), Integer (16-bit register), Float or Long. Each variable can be assigned a 15-character label that can include spaces and other ASCII characters. This label is then displayed as a comment alongside each variable when the function block is called using a Call function block.

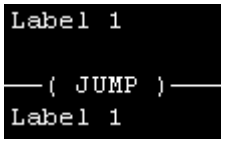
Return Variable Type: None or Boolean (True or False). If defined as None, it is an input function. If defined as Boolean it is an output function.



Return

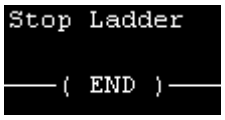
Transfers ladder logic processing back to the next block after the *Call Function Block*. If the Return block is used with an output function, the return value must be *none*; if the Return block is used with an input function, the return value must be *True* or *False*. As used in the topic Example – Time Based Rolling Averages.

Return Value: (None, True or False) The result returned by the function.



Jump

Jump to label *Label1*. Can jump forwards and backwards. Labels are defined in the comment field of the first block on the rung and must have the format: *LabelName:* ie. the label name followed by a colon. Labels cannot be defined in a *Comment Field* block. As used in the topic Example – Time Based Rolling Averages.



End

Stops processing of the ladder at that point. This can improve the scan rate of ladder logic by preventing the unnecessary scanning of ladder logic located after the End block. An End block is automatically inserted into the compiled output file just before the definition of the first function block.

Caution! An End block must NEVER be used *within* a function block as it will prevent the function block from correctly finishing and may cause the RTU to behave unpredictably.

Ladder Logic - P.I.D. Block

The PID block is used to monitor a process variable (eg. flowrate) and compare it to a setpoint (eg. desired flowrate). According to the difference between the actual value and the setpoint value, the PID block sets a Control Variable (eg. valve position) to reduce the error. The control variable is gradually changed until the desired setpoint is achieved within the deadband limits.

The rate of change of the Control Variable is configurable so that for delicate processes, the output rate of change can be small and for robust processes requiring quick responses, the output rate of change can be high.

The example shown below is used to control a valve to achieve a setpoint flowrate. The output of the PID is stored in #R1 (which is used to control the valve position). The actual valve position is read from #AI14.2, and the required flowrate is stored in #R2. The PID block uses reverse action and a proportional gain of 1 so that a drop of say 5% in flowrate (#AI14.2) will result in an increase of 5% in valve position (#R1). In addition an Integral Factor of 0.1 units/min is used so that the valve position will be increased at a rate of 0.5% each minute until the flowrate is within 1% (327) of the setpoint. As the PID block is in Auto mode, the Raise and Lower parameters are not used.

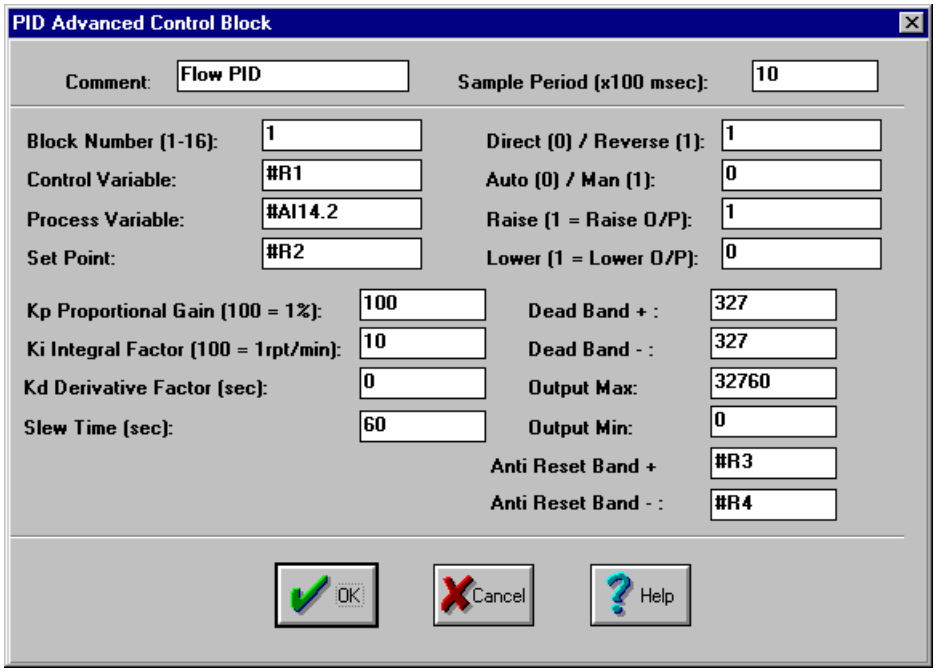

PID Advanced Control Block
Comment: Flow PID
Sample Period (x100 msec): 10
Block Number (1-16): 1
Direct (0) / Reverse (1): 1
Control Variable: #R1
Auto (0) / Man (1): 0
Process Variable: #AI14.2
Raise (1 = Raise O/P): 1
Set Point: #R2
Lower (1 = Lower O/P): 0
Kp Proportional Gain (100 = 1%): 100
Dead Band + : 327
Ki Integral Factor (100 = 1rpt/min): 10
Dead Band - : 327
Kd Derivative Factor (sec): 0
Output Max: 32760
Slew Time (sec): 60
Output Min: 0
Anti Reset Band + #R3
Anti Reset Band - : #R4
OK
Cancel
Help

Figure: Example PID Advanced Block.
Note: the standard PID block does not have the Anti Reset Band parameters.

Comment: A 12-character description.

Block Number (1-16): The PID block number in the ladder. Up to 16 PID blocks can be configured in ladder logic.

Control Variable: (16-bit register) The *output* of the PID block used to control a process to produce the desired Set Point. This can be a hardware register or a local register.

Process Variable: (16-bit register) The *input* that is monitored by the PID block. Process error is determined from the difference between the Process Variable and the Set Point. The process variable is usually an analog input which is stored in the RTU as a number in the range 0-32760 (equivalent to 0.00 - 100.00%).

Set Point: (16-bit register or constant [0 to 32767]) This is the desired process result. The PID output will be continually adjusted until the Process Variable reaches this setpoint. The setpoint is given the same units or scaling as the Process Variable and so if an analog input is being monitored the setpoint should have a range of 0-32760 (0.00 - 100.00%).

Kp Proportional Gain (100 = Gain of 1): (16-bit register or constant [0 to 32767]) The proportional response is the proportional change in the control variable (the output) in response to a change in the process variable (the input). Hence, Proportional Response = Kp/100 x (input change). Eg. If Kp=100 then a change of 3 units in the input will result in a proportional response of 3 units in the PID output. Similarly, if Kp=200 then a change of 3 units in the input will result in a proportional response of 6 units in the PID output.

Ki Integral Factor (100 = 1rpt/min): (16-bit register or constant [0 to 32767]) The integral response is the rate of change of the process variable (the output) that will occur after the proportional change in order to reduce the error between the Set Point and the process variable (the input). Integral response is defined as: Integral Response [units/min] = Ki/100 x (Input Error). Eg. If Ki=10 and the error is 200, then the integral response = 10/100 x 200 = 20 units every minute. This means the output will be changed unit by unit up to a total of 20 units over the minute interval (the output will be changed by 1 every 3 seconds).

Kd Derivative Factor (min): (16-bit register or constant [0 to 32767]) The PID task keeps track of the last three errors (ie. the difference between the input and the setpoint) and according to the rate of change in the error will take the appropriate action. This usually results in a dramatic change in the output and so Kd is not used in many applications. Derivative response is defined as: Derivative Response = Kd/100 x 600/(Sample Period) x (Error Change). Eg. If Kd=100, Sample Period=10 (1 second) and the error increased by 1 in the last second (since the PID block was last processed), then the integral response = 100/100 x 600/10 x 1 = 60 units. To disable this function leave Kd set to 0. Note: a derivative factor is not recommended for any fragile process.

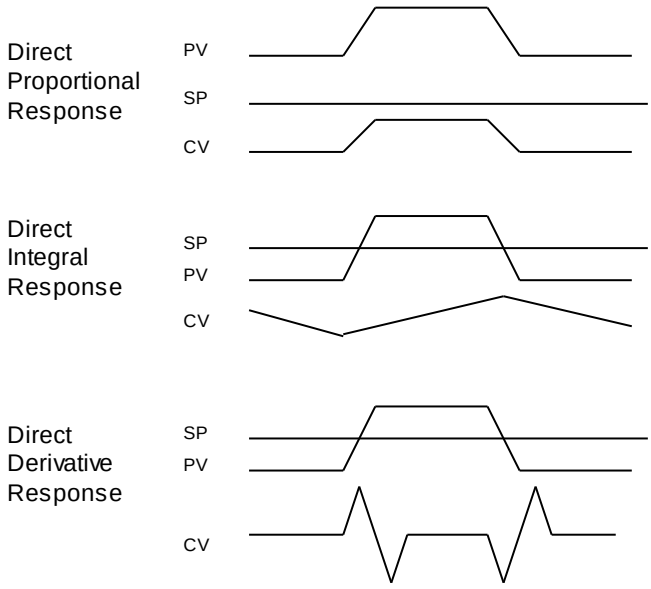
Slew Time (sec): (0=freeze output or 1 to 32767) Slew time is the total amount of time it takes for the output to go from *Output Min* to *Output Max* or vice versa. This should be set to match the slew rate of the controlled actuator or other device to prevent damage caused by the PID control variable changing too rapidly.

Sample Period (x 100ms): (0 to 32767) How often the PID block is processed. One unit = 0.1 seconds (100 ms). It is recommended that a minimum setting of 10 be used (ie. Sample Period = 10 x 100 ms = 1 sec) to allow enough time for the RTU processor to manage communications and other functions.

Direct (0) / Reverse (1): (Bit) This determines whether the Control Variable (the output) will be increased or decreased for a given error between the Set Point and the Process Variable (the input). The direct action taken for the various conditions is detailed below:

Control Type	**Process Variable (PV) (Input) Condition**	**Control Variable (CV) Direct Response***
Proportional	PV increases	CV increases
	PV decreases	CV decreases
Integral	PV > Set Point	CV increases
	PV < Set Point	CV decreases
Derivative	Rate of change of PV increases	CV increases
	Rate of change of PV decreases	CV decreases

* For Reverse Action PID, the control variable response is reversed



Auto (0) / Man (1): (Bit) When in manual mode, the PID block uses the Raise and Lower parameters to control the output. When in auto mode, the PID block adjusts the Control Variable until it is within the Deadband settings of the Set Point.

Raise (1 = Raise O/P): (1=enabled, 0=disabled) Only used in manual mode. When enabled, the output is continuously raised at the rate specified by the Slew Time. *Raise* and *Lower* should not be enabled at the same time as the output will be unpredictable. If neither are enabled the output will remain constant.

Lower (1 = Lower O/P): (1=enabled, 0=disabled) Only used in manual mode. When enabled, the output is continuously lowered at a rate specified by the Slew Time.

Deadband +: (0 to 32767) The allowable positive error between the Process Variable (the input) and the Set Point. When used with analog inputs this is a number in the range 0 to 32760. Error is calculated from Process Variable minus Set Point and so positive error occurs when the process variable is above the setpoint. The PID block will continue to change the output until the positive error is less than or equal to this setting. Eg. For a 1% Deadband for an analog input, 327 would be used.

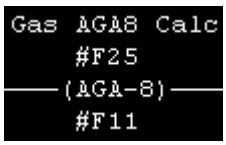
Deadband -: (0 to 32767) The allowable negative error between the Process Variable (the input) and the Set Point. When used with analog inputs this is a number in the range 0 to 32760. Error is calculated from Process Variable minus Set Point and so negative error occurs when the process variable is below the setpoint. The PID block will continue to change the output until the negative error is less than or equal to this setting. Eg. For a 1% Deadband for an analog input, 327 would be used.

Output Max: (0 to 32767) The maximum allowable PID output. If the PID output is being used to set an analog output, then this parameter should be set in the range 0-32760 corresponding to 0-100%.

Output Min: (0 to 32767) The minimum allowable PID output. If the PID output is being used to set an analog output, then this parameter should be set in the range 0-32760 corresponding to 0-100%.

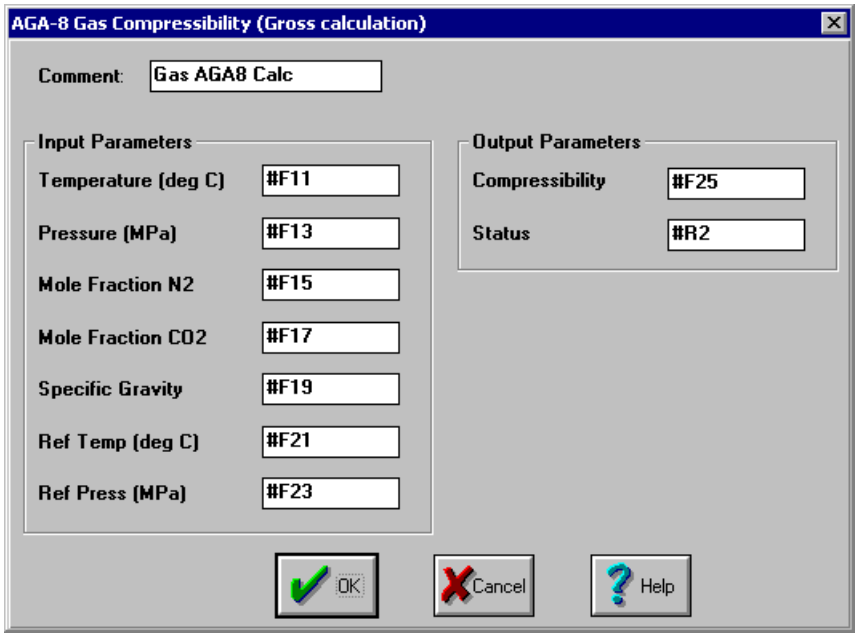
Anti Reset Band +/-: [PID Advanced block only] (0 to 32760) The anti-reset windup feature inhibits the integral action until the Process Variable (PV) is within the Anti Reset Band thus reducing overshoot on start-up. The anti reset function will be disabled if the Anti Reset Band + value is set to 0. While the PV is less than or equal to Anti Reset Band -, the integral action function of the P.I.D. loop will be disabled. While the PV is greater than or equal to Anti Reset Band +, the integral action function of the P.I.D. loop will be disabled. The integral action function of the P.I.D. loop will be enabled at all other times.

Ladder Logic - AGA-8



AGA-8 Gas Compressibility (Gross calculation)

Uses the American Gas Association standard AGA-8 for calculating gas compressibility. Requires driver *AGA8.Dxx*.



Comment: A 12-character description.

Temperature (deg C): (Float) Gas temperature.

Pressure (MPa): (Float) Gas pressure.

Mole Fraction N2: (Float) Mole fraction of Nitrogen in the gas mixture.

Mole Fraction CO2: (Float) Mole fraction of Carbon Dioxide in the gas mixture.

Specific Gravity: (Float) Specific gravity (relative density) of the gas mixture.

Ref. Temperature (deg C): (Float) Reference temperature.

Ref. Pressure (MPa): (Float) Reference pressure.

Compressibility: (Float) AGA-8 compressibility factor of the gas.

Status: (Local register) AGA-8 calculation status. The following channels are used:

- Ch 1: calculation error
- Ch 2: pressure out of range error (allowed range is 0 to 12 MPa)
- Ch 3: temperature out of range error (allowed range is -8 to 62 ^{o}C)

Example:

Temperature	0°C
Pressure	0.6894757 Mpa
Mole N2	0.002595
Mole CO2	0.005956
Specific Gravity	0.581078
Reference Temperature	15.56 °C
Reference Pressure	0.101560 Mpa

Result: Compressibility=0.982387, Status=0.

AGA-7 Gas Flow

This formula uses the compressibility output from the AGA-8 block and can be written in ladder to evaluate the AGA-7 gas flow (note: an example configuration is available from http://helpdesk.servelec-semaphore.com).

Qv = Qf x P/Pgr x Tgr/T x Zb/Zf

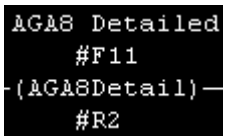
Qv = Volumetric flow, cubic metres/second
Qf = Flowrate at flowing conditions, cubic metres/second
P = Pressure, MPa
Pgr = Reference pressure for specific gravity, MPa
T = Temperature, °C
Tgr = Reference temperature for specific gravity, °C
Zb/Zf = Compressibility factor (output from AGA-8 block)

AGA-3 Gas Flow

The AGA-3 gas flow can be obtained by writing the following formula in ladder logic (note: an example configuration is available from http://helpdesk.servelec-semaphore.com):

Qv = N1 * Cd * Ev * Y * d^2 * sqrt (dP / p)

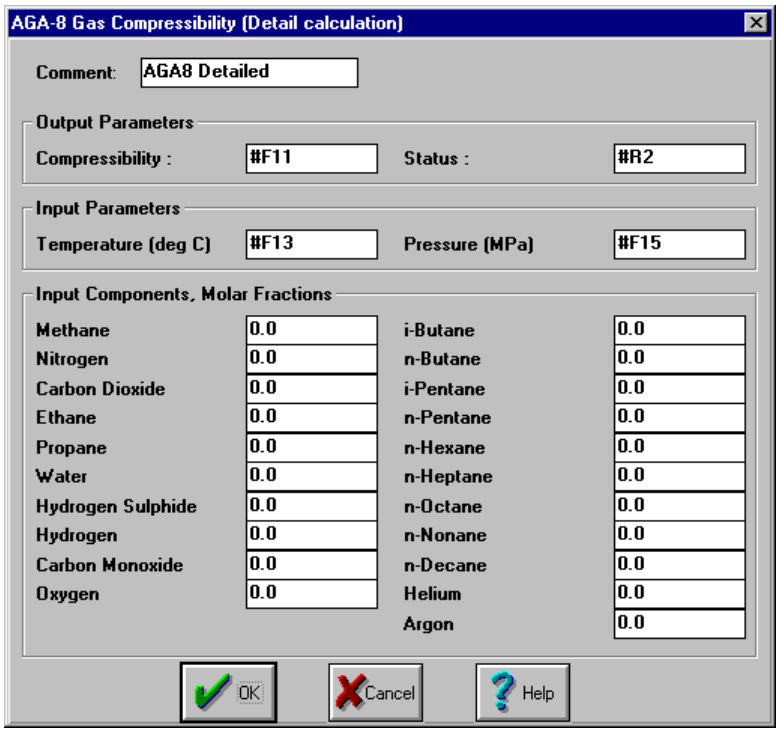
Qv = Volumetric flow
N1 = Unit conversion factor (orifice flow)
Cd = Orifice plate coefficient of discharge
Ev = Velocity of Approach Factor = 1 / sqrt (1 - b^4)
b = Orifice bore to meter tube diameter ratio = d / D
d = Orifice plate bore diameter = dr * (1 + a * (Tf - Tr))
dr = reference orifice plate bore diameter at Tr
a = linear coefficient of thermal expansion
Tf = temperature of fluid at flowing conditions
Tr = reference temperature (eg. from analog input channel)
D = Meter Tube internal diameter = Dr * (1 + a * (Tf - Tr))
Dr = reference meter tube internal diameter at Tr
Y = Expansion factor
dP = Orifice differential pressure (eg. from analog input channel)
p = Density of fluid at flowing conditions



AGA-8 Gas Compressibility (Detailed calculation)

Uses the American Gas Association standard AGA-8 for calculating gas compressibility using the detailed characterization method. Requires driver *AGA8DET.Dxx*.

The driver is based entirely on *Compressibility Factors of Natural Gas and Other Related Hydrocarbon Gases, AGA Transmission Measurement Committee Report No. 8, Second Edition, November 1992*.



Comment: A 12-character description.

Compressibility: (Float) AGA-8 compressibility factor of the gas.

Status: (Local register) AGA-8 calculation errors and warnings. The following channels are used:

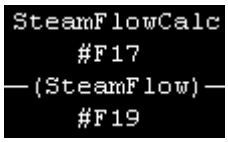
- Ch1: Internal calculation error. The AGA8 Detail calculation is a fairly complicated nonlinear calculation that includes a couple of iteration loops (ie. it repeats a calculation many times until the result converges to a solution). The AGA8 driver limits these loops to a maximum of 100 iterations each, to limit the time taken to perform the calculation. An *Internal calculation error* indicates that after 100 iterations the result was still varying slightly (a valid result is still returned by the AGA8 calculation block). The reason for the variation may be that the input parameters are close to the limits specified in the AGA8 Detailed specification. For further details about the parameter limits and the tolerance levels, please refer to the actual AGA8 Standard. If very precise accuracy is required from the result, the *internal calculation error* bit can be used as a warning flag, otherwise it can be ignored.
- Ch2: Pressure out-of-range error (allowed range is 0 to 280 MPa)
- Ch3: Temperature out-of-range error (allowed range is -130 to 400 ^{o}C)
- Ch4: Firmware driver error
- Ch5: Gas components outside *normal* range
- Ch6: Gas components outside *expanded* range

Temperature (deg C): (Float) Input variable representing gas temperature.

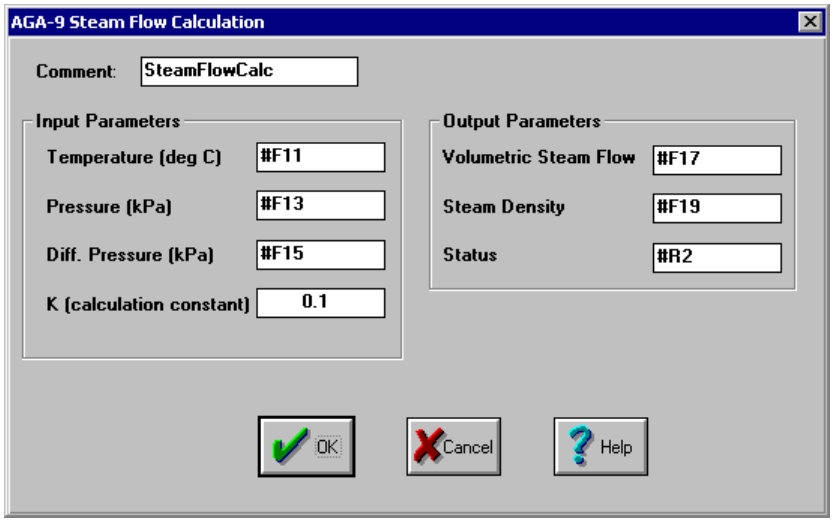
Pressure (MPa): (Float) Input variable representing gas pressure.

Input Components, Molar Fractions: These parameters represent the relative proportions of each component in the gas mixture. They can be entered in any units (percentages, fractions, etc).

Ladder Logic - AGA-9 Steam Flow Calc



Calculates steam flow using the AGA-9 standard and the published tables of superheated steam density. Requires driver *AGA9.Dxx*.



Volumetric steam flow is calculated using the formula:

Qv = K x sqrt(F) x sqrt(dP)

dP = differential pressure
F = steam density (from steam tables; function of T,P)
T = temperature
P = pressure
K = calculation constant = Fna x K' x Fra x Fm x Faa x F1 x D^2
Fna = units correction factor
K' = flow coefficient
Fra = Reynolds number correction
Fm = manometer correction factor
Faa = thermal expansion factor
Fl = gauge location factor
D = pipe diameter

Comment: A 12-character description.

Temperature (deg C): (Float) Steam temperature.

Pressure (kPa): (Float) Steam pressure.

Diff. Pressure (kPa): (Float) Differential steam pressure.

K (calculation constant): (Float) Calculation constant K (as detailed above).

Volumetric Steam Flow: (Float) AGA-9 volumetric steam flow (cubic metres/second).

Steam Density: (Float) AGA-9 calculated steam density (kg/cubic metre).

Status: (Local register) AGA-9 calculation status. The following channels are used:

- Ch1: Calculation error (indicates steam is saturated)
- Ch2: Pressure out-of-range error (allowed range is 0 to 7000 MPa)
- Ch3: Temperature out-of-range error (allowed range is 100 to 700^{o}C)

Examples:

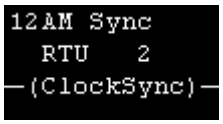
Temperature	190^{o}C
Pressure	2000 kPa
Differential Pres.	20 kPa
K	0.05

Result: Volumetric Steam Flow=0, Steam Density=0, Status=1 (saturated)

Temperature	200^{o}C
Pressure	1000 kPa
Differential Pres.	100 kPa
K	0.1

Result: Volumetric Steam Flow=2.20318, Steam Density=4.854, Status=0

Ladder Logic - Clock Synchronization



Synchronizes the RTU's real-time clock. There are two modes of operation:

- Single RTU Sync: Forces the real-time clock of the target RTU to match the local RTU.
- Global RTU Sync: Sends a global command to synchronize all RTUs that are connected to the same comms port as the target RTU.

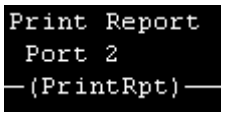
In both modes of operation, the communication delay is first measured between the local RTU and the target RTU. The clock synchronisation message is then adjusted to compensate for this delay. Note: if communications to the target RTU fail, a global clock synchronization is not carried out.

Comment: A 12-character description.

RTU # (1-255): Target RTU to be synchronized and to be used for calculating the communication delay.

Global Sync Command? Indicates whether to issue a single or global sync command. Do not use a Global Sync Command when the target RTU is indirectly connected (ie. via a store-and-forward RTU) as the longer communication delay will be added to the clocks of all the directly connected RTUs when the global clock synchronization message is sent.

Ladder Logic - Report Printer



Prints a text file to a serial printer. The text file may contain RTU variables. The text file is compiled with the ladder logic and stored in the RTU. Requires driver *REPORT.Dxx*.

Comment: A 12-character description.

Filename: The filename is automatically generated and a text file is created with that name. The file can then be edited and various variables and text added as detailed below. When the ladder logic is compiled, the text file is included with the compiled code.

Port: (1-16) The serial port to print from.

Report File: The report file has 3 sections: Text, Variables and End.
TEXT: This section contains all the text and variables that will be sent to the serial printer line by line. A variable is included by writing a % followed by an integer (1-65535) and then declaring the variable under the Variables section. Eg: Pump 1 Starts Today: %1 (%1 is then declared under the variables section)
VARIABLES: All the live variable information is declared in this section. Variables may include local registers (#R), network registers (#N), system registers (#Y) and any other parameter that can be used in ladder logic. The number and order of Variable declarations must match the number and order of variables (%n) used in the Text section. There is no error checking.
END: Denotes the end of the report file.

Example report file:

```
TEXT:
Pump 1 Starts Today:    %1
Pump 1 Status:          %2

PC-1 Battery Status:    %3
RTU2 Comms Status:      %4

VARIABLES:
#R1         %05i
#R2.1       %b    STOPPED   RUNNING
#DI13.3     %b    LOW       OK
#YLST2.1    %b    OK        FAIL

END
```

Each variable is defined using the following layout:

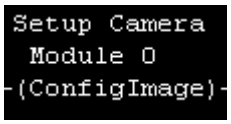
Address Format Additional strings

Where:
Address: Any variable as used in ladder logic (eg. #R1, #YDIAG.1).
Format: A string defining the display format of the variable. The format string has the following structure:
(Note: [ ] denotes an optional parameter)

% [Flags] [Width] [.Prec] [_dp] [l] Type

Format parameter	Options	Description
Flags	-	Left justified
	+	Value starts with a + or -
	#	If type is o, value begins with a 0 If type is x or X, value starts with 0x If type is e, E or f, value will have a decimal point. If type is g or G, value will have a decimal point and trailing zeros will not be removed.
	<Blank>	If negative, value starts with -
Width	n	At least n characters are printed. The value is padded with blanks.
	0n	At least n characters are printed. The value is padded with leading zeros.
.Prec	.0	For e, E, f types, no decimal point is printed.
	.n	n characters or n decimal places are printed. For e, E, f, g, G types, the last digit printed is rounded.
	<Blank>	Defaults to .1 for d, i, o, u, x, X types. Defaults to .6 for e, E, f types. Displays all significant digits for g, G types.
_dp	_n	Decimal places. A decimal point is inserted in the output with n digits following. For d, i, u types only.
l (long)	l	Displays d, i, o, u, x, X types as long values.
Type	d	Signed decimal integer
	i	Signed decimal integer
	o	Unsigned octal integer
	u	Unsigned decimal integer
	x	Unsigned hexadecimal integer using lower-case letters a-f
	X	Unsigned hexadecimal integer using capital letters A-F
	f	Floating point signed value of the form [-]dddd.dddd
	e	Floating point signed value of the form [-]d.dddd or e[+/-]ddd
	E	Same as e, but with capital E for exponent
	g	Floating point signed value in either the f or e form based on given value and precision
	G	Same as g, but with E for exponent if e format is used
	b	Bit string. Displays the first string if the bit is False or displays the second string if the bit is True. Strings must be separated by either tabs or spaces.
	a	Animated string. If the value of the variable is 0, the first string is printed. If the value is 1 the second string is printed, etc. If the value exceeds the number of given strings the last string is printed. Strings must be separated by either tabs or spaces.

Ladder Logic - Image Monitoring Functions



Configure Image Parameters

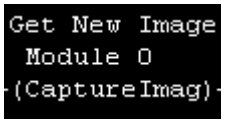
Configures an image channel before it is used for the first time or when changing to a new channel. PC-1/CP-1 requires driver *IMAGExx.DRV* to support an MC-xx with an Image Capture option board. Please see the topic Example - Kingfisher Images.

Slot Address: (0-64) The slot address of the module with the image option board. Set to *0* for a CP-10/11 or set to the slot address for an MC-10/11/12.

Channel Number: The image capture input channel (1-4). A port 2 image board uses channels 1 and 2. A port 3 image board uses channels 3 and 4. The top connection on each image board corresponds to the lower channel number.

Resolution: 0 = PAL, large (352W x 288H pixels), 1 = PAL, medium (176W x 144H pixels), 2 = PAL, small (88W x 72H pixels), 16 = NTSC, large (320W x 240H pixels), 17 = NTSC, medium (160W x 120H pixels), 18 = NTSC, small (80W x 60H pixels). Note: setting a resolution of small is the same as setting a medium resolution.

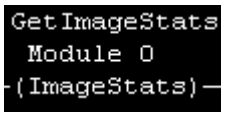
Quality Factor: The JPEG image quality (1-100). 1 = lowest quality (2-3 KB per image) and 100 = highest quality (10-15 KB per image). A quality factor of at least 80 (approx. 10KB per image when resolution=0) is recommended.



Capture Image

Captures a single image. The image is then added to a circular buffer of images. Before an image can be captured, the image channel must first be setup using the Configure Image Parameters block (as detailed above). PC-1/CP-1 requires driver *IMAGExx.DRV* to support an MC-xx with an Image Capture option board.

Slot Address: (0-64) The slot address of the module with the image option board. Set to *0* for a CP-10/11 or set to the slot address for an MC-10/11/12.



Get Image Buffer Statistics

Monitors the RTU's image memory buffer. PC-1/CP-1 requires driver *IMAGExx.DRV* to support an MC-xx with an Image Capture option board.

Slot Address: (0-64) The slot address of the module with the image option board. Set to *0* for a CP-10/11 or set to the slot address for an MC-10/11/12.

Total Images: (Optional local register) The total number of images that are stored in the RTU. Once the RTU's image buffer is full, this number will not change.

Unread Images: (Optional local register) The total number of images that have never been read by Image Manager or any other software.

Next Image Number: (Optional local register) Image number to be assigned to the next image captured.

Image Pointer: (Optional Local register or FFFF) A local register that points to a particular image in the buffer. Used by Image Manager to keep track of the last image uploaded from the RTU.

Total Images After Pointer: (Optional local register) The total number of images in the buffer after the image specified by Image Pointer above. When Image Pointer is set to FFFF (or to a value outside the range of images in the RTU), Total Images After Pointer will return the total number of images in the RTU (the same as Total Images above).

First Image After Pointer: (Optional local register) The first image after the image specified by Image Pointer above. When Image Pointer is set to FFFF (or to a value outside the range of images in the RTU), First Image After Pointer will return the image number of the oldest image in the RTU.

6. Ladder Logic Examples

To create or edit ladder logic, select **Logic, Edit**.

Examples in this chapter:

General	Initialising Variables, Timer Flags, Counting Pulses, Hours Run, Flow Totalisation, Rolling Over Counters, Counting Comms Fails, Synchronising RTU Clocks, RTU Diagnostics
Exception Reports	Analogs, Digitals, Sending
Event Logging	Creating Event Logs
Polling	Data, Event Logs, Advanced Polling
Communication Devices	Modems, Radios, Satellite Phones, GPRS Modems, RS485, Private Line
SMS / Paging	SMS Pager Messages, SMS Messages With Variables
Images	Image Capture
Low Power	Low Power Mode
Advanced	Indirect Addressing, Rolling Averages
Ladder Logic	Downloading, Compare Ladder Version In RTU, Debugging

Note: The examples in this chapter can be downloaded from http://helpdesk.servelec-semaphore.com.

Example - Initialising Variables On First Scan

Variables can be initialised after ladder logic is downloaded by using the logic below. To initialise variables after a warm start or power reset or after downloading the SDB file, use #YSYS.SCAN1 instead of #YSYS.ENABLE.

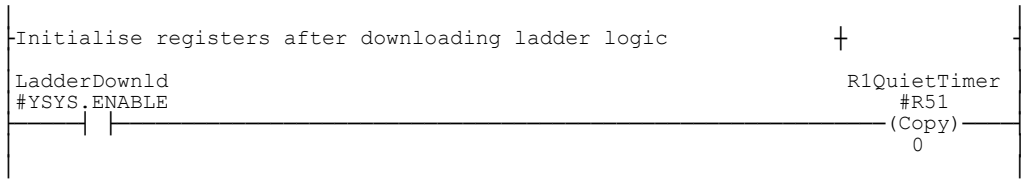


Figure: Initialising variables on the first ladder scan

Example - Timer Flag

An RTU has 64 timer registers that can be used once each in ladder logic. Although the amount of timer registers may appear to be low, each timer register can be used to regularly set and reset a register bit. The register bit can then be used an unlimited number of times in the ladder to rollover totals or to regularly copy data into registers etc. In addition to timers, there are also a number of periodic system registers which can also be used an unlimited number of times. Please see the appendix RTU Data, System Registers, Clock Registers. The *3.6 Sec Flag* below is true once every 3.6 seconds and is true for one scan of the ladder.

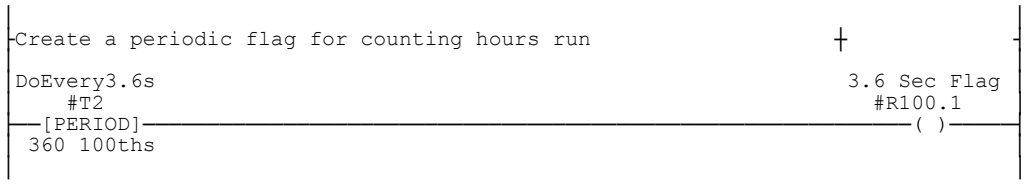


Figure: 3.6 Second Periodic Flag (used for counting hours run)

Example - Counting Pulses And Starts

Counting Pulses Using A Standard Digital Input (DI-1, IO-x)
An RTU is capable of counting input pulses up to a rate of at least 10 Hz. The actual pulse rate that the RTU can count depends on how often it is able to scan its ladder. Since pulses are counted by counting the rising or falling edges of digital inputs, the ladder needs to be scanned fast enough to allow the RTU to register the pulse in the ACTIVE and in the INACTIVE states. Assuming that a pulse has a 50% duty cycle then the maximum pulse rate that can be counted is half the maximum ladder scanning rate (as displayed by the RTU Status). An example of an acceptable pulse input is shown below.

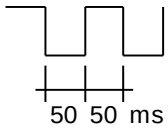


Figure: Acceptable Pulse Input That The RTU Is Able To Count.

Figure 5.3b shows how to count pulses using a rising edge trigger. Every time there is a new pulse, the *Pulses Today* register is incremented.

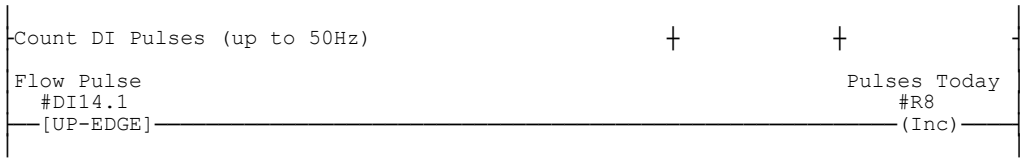


Figure: Counting Standard DI Pulses

Shaft Encoder - Quadrature Pulse Counting

A shaft encoder has two pulse outputs. Each time the level changes, a pulse is generated on each output. Depending on which output pulsed first, the direction of the level change can be determined. By beginning with a default level (eg. say 50%), the total can be incremented or decremented according to whether the level change is positive or negative. In the example below, #R15 contains the Shaft Encoder Level. The level is set to a default of 1000 after a warm start and is incremented or decremented depending on the order of the pulses from LP-1/2/3 digital channels 1 and 2. Note: the accuracy of this type of quadrature counting depends on the RTU scanning fast enough to capture each pulse (a scan rate of at least 100 times a second is recommended). For greater accuracy, a DI-10 module can be used.

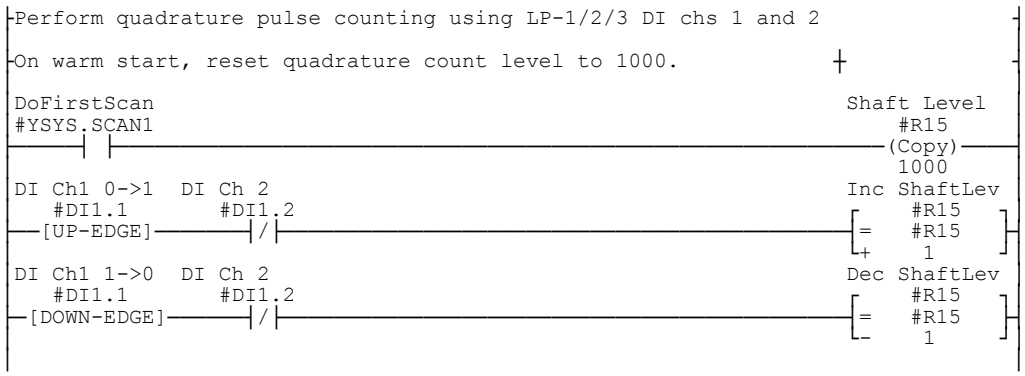


Figure: Quadrature Pulse Counting From a Shaft Encoder

Counting Pulses Using A DI-5

A DI-5 Counter Module automatically keeps pulse totals for its first four digital input channels. For fast pulse rates (up to 10kHz), the pulse totals will reach the maximum value of 65535 very quickly. To prevent a pulse total from overflowing, it can be rolled over every 1000 pulses as shown below. In the example, #AI14.2 is the number of digital input 1 pulses (0-999) and #R14 is the number of thousands of pulses (0-65535 k) for a DI-5 module in slot 14.

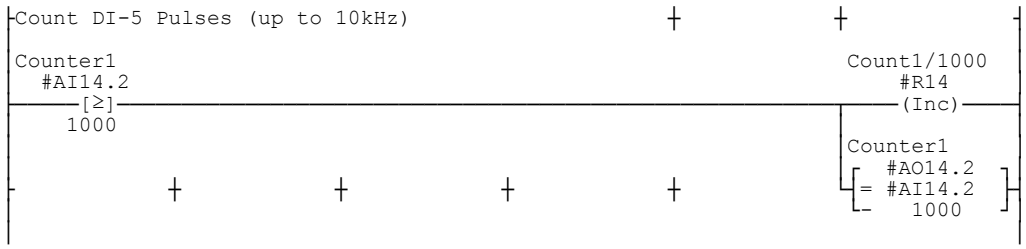


Figure: Counting DI-5 Pulses (up to 10kHz)

Counting Starts

Counting starts is exactly the same as counting pulses using a standard digital input. The start and stop signals are like a very slow pulse. The ladder below shows how to count starts using a rising edge trigger. Every time there is a new start, the *P1 StartsTdy* register is incremented.

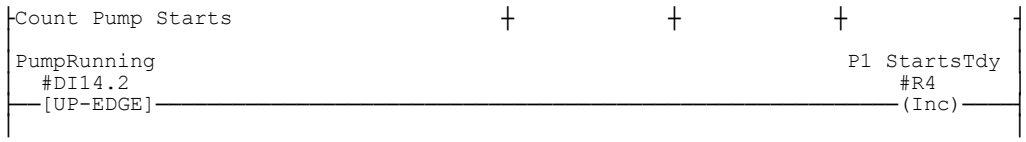


Figure: Counting Pump Starts

Example - Hours Run

The example below records how long pump 1 has been running for in decimal hours. Every 3.6 seconds (0.001 Hrs) the ladder checks if pump 1 is running and if it is, *P1 HrsRunTdy* (#R6) is incremented. Local register #R6 then contains the number of 0.001 hour intervals that the pump has been running for ie. 0-65,535 = 0-65.535 Hrs. *P1 HrsRunTdy* is rolled over at midnight (as shown in section 5.6 - Rolling Totals Over At Midnight) so that the total will not overflow after a few days. The ladder makes use of the *3.6 Sec Flag* from the topic Example - Timer Flag.

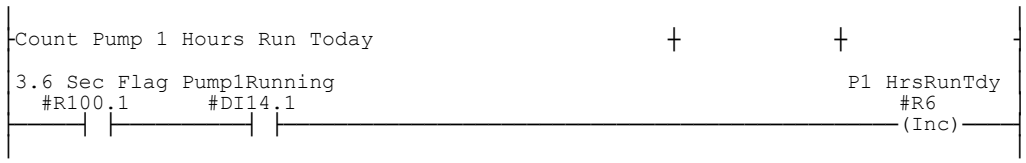


Figure: Counting Hours Run Today

Example - Flow Totalisation

The following example shows how to accumulate a flow volume from a flowrate analog input. For this example, the flowrate engineering units are 4-20mA=0-100 L/s. Each second, the number of litres that have flowed (*FlowLastSec*, #R10) is calculated by dividing the analog input by 32760 (the raw analog input range) and then multiplying by 100 (the high limit of the engineering units). This number of litres is then added to the *FlowTdy(L)* total (#R11). When *FlowTdy(L)* equals or exceeds 1000 litres, it is rolled over into another register *FlowTdy(kL)* (#R12).

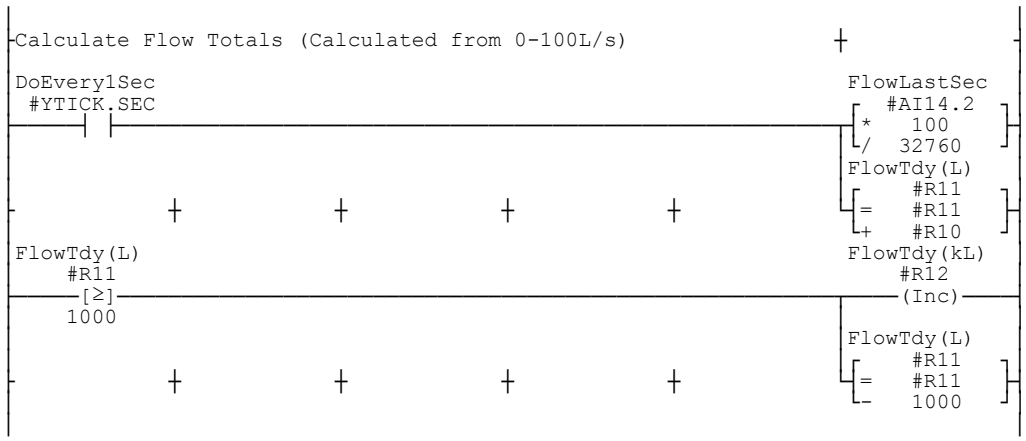


Figure: Flow Totalisation

Example - Rolling Totals Over At Midnight

When the real time clock reaches midnight, or the RTU detects that the day has changed, a *12AMRollover* flag can be set. This flag stays true for 1 ladder scan and can be used to roll over totals at midnight.

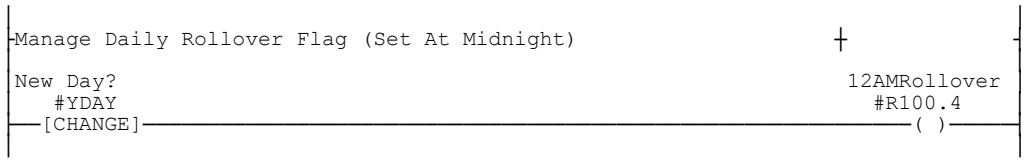


Figure: 12AM Rollover Flag

The example shown below rolls over *P1 StartsTdy* by copying the total to *P1 StartsYes* register and then resetting *P1 StartsTdy* to zero. Note: a number of rollovers can be combined into a single Multi-Copy block.

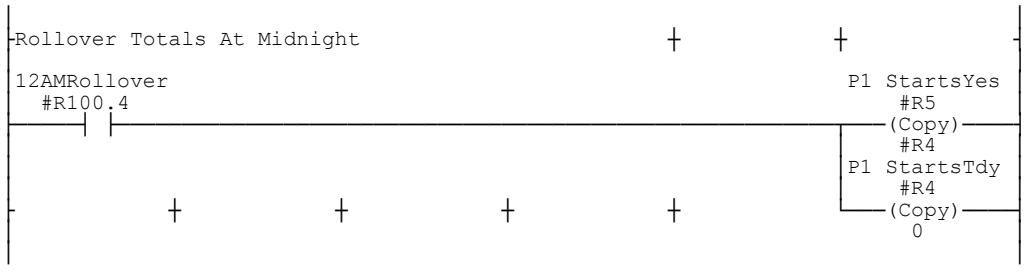


Figure: Rolling Over Totals At Midnight

Example - Exception Reporting Analogs

Analog values can be exception reported to the master RTU when there has been a percentage change (of the analog range) from the last reported value. This is done by using two registers, a constant and an analog input. The constant is used to specify the amount the analog value must change by before an exception report is generated. The registers are used to store the last reported value plus the constant and the last reported value minus the constant. When the analog value moves above or below these register values, an exception report is generated and the registers are updated.

The lower register limit must be checked if it is negative as a negative number is stored as a very large integer value and can cause continuous exception reports as the low limit comparison block is always true.

The constant to use is calculated as percentage of the analog or register range. For analog inputs which have a range of 0-32760 (32767 for an AI-10), a 1% change is represented in the RTU by a change of about 327. Similarly, a 5% change is represented in the RTU by a change of 1638 (0.05 x 32760).

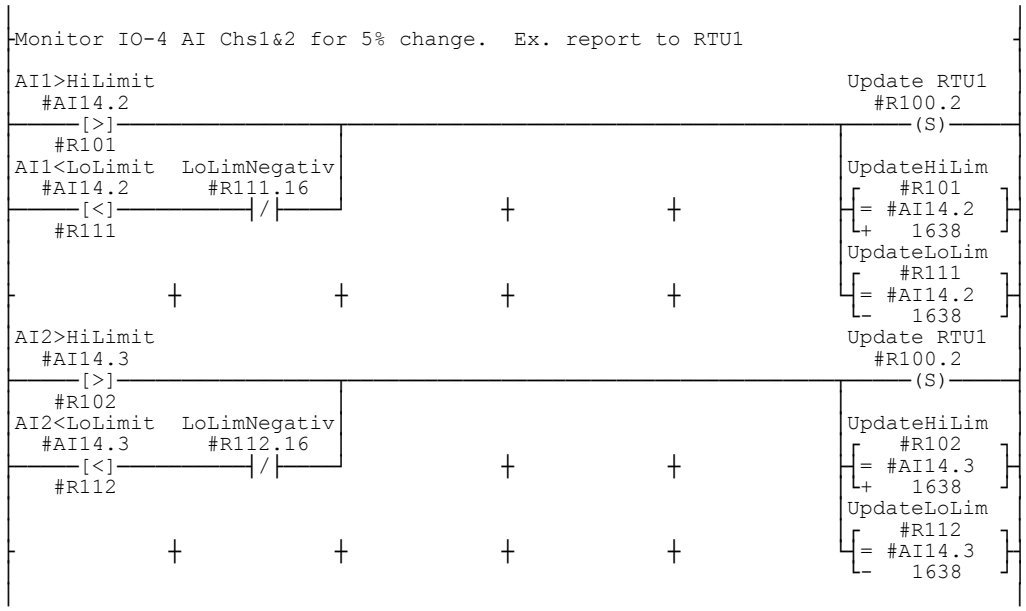


Figure: Exception Reporting IO-4 Analog Channels 1 and 2.

Example - Exception Reporting Digitals

An exception report can be generated when a single digital bit changes state or when any of the 16 channels in a hardware or local register change state.

The example below shows how to exception report a PC-1 mains power fail. The 30 second on-delay is used to prevent false exception reports caused by the discharge LED flickering ON and OFF (which can happen when the battery is fully charged or is not connected). After 30 seconds of continuous battery discharge, the mains power to the PC-1 is said to be OFF.

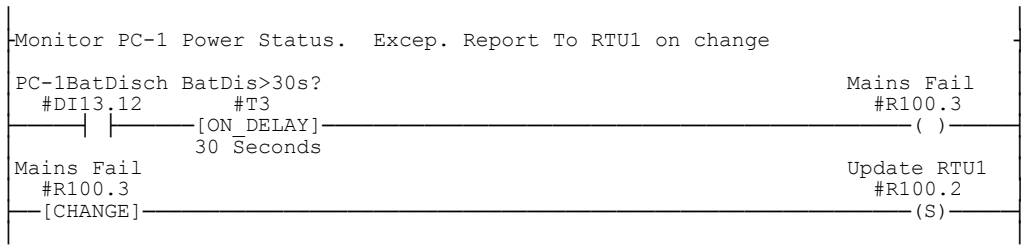


Figure: Exception Reporting A Single DI Channel.

The example below shows how to monitor all 16 channels from a register or from a DI module and to exception report when any channel changes state.

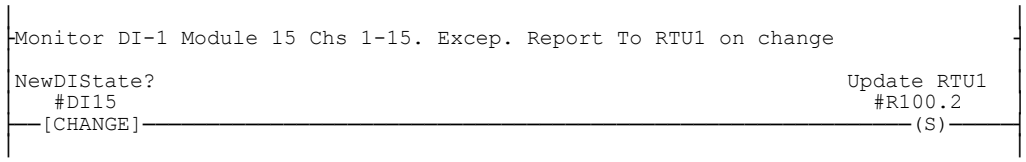


Figure: Exception Reporting 16 DI Channels

The example below shows how to monitor specific channels from a register or from a DI module. Parameter 2 in the AND block is a constant that has all the channels set to 1 that need to be monitored. Eg. to monitor channels 5-16, the hexadecimal constant 16#FFF0 (65520) is used (for a description of hexadecimal numbers and masking please see the appendix - Hexadecimal Numbers).

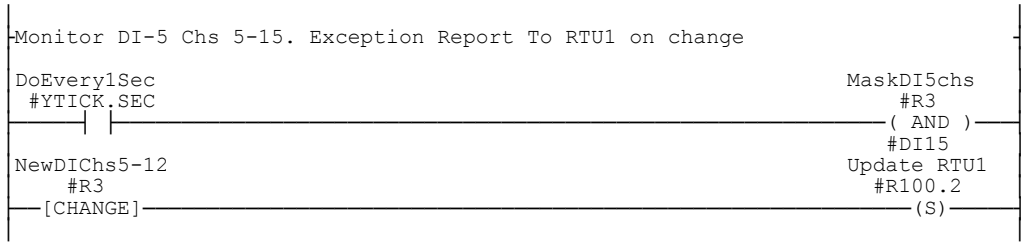
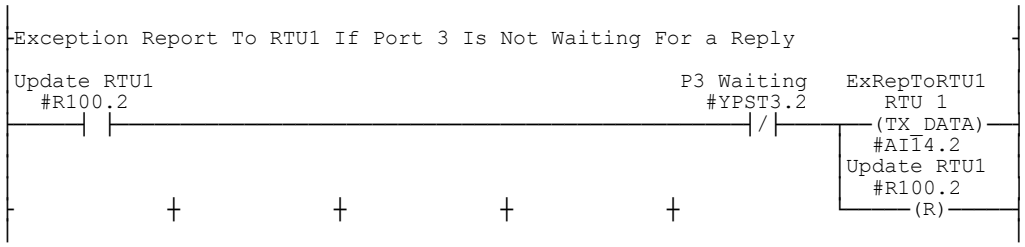


Figure: Exception Reporting Specific DI Channels.

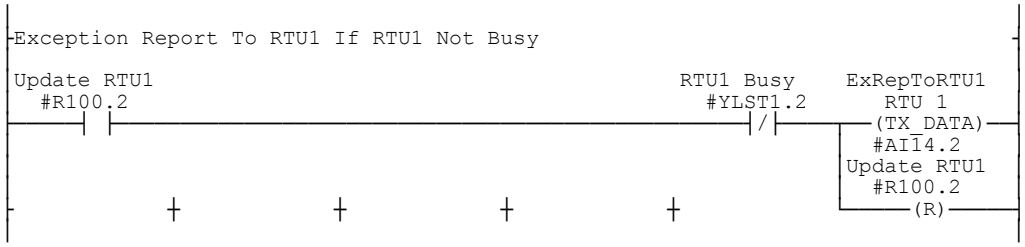
Example - Sending The Exception Report

An RTU usually has a number of inputs and conditions that can trigger an exception report. Rather than exception report every change separately, it is better if all these conditions set a common flag that causes a single exception report to be sent containing all the RTU data. As shown in the previous sections for exception reporting analogs and digitals, the *Update RTU1* flag was set when an exception report was required. An example of ladder logic used to send the exception report is shown below (for all types of comms except Ethernet).



Exception Reporting Using Ethernet

It is possible to communicate with multiple RTUs at the same time using one Ethernet port. Therefore, instead of checking if the port is busy (waiting for a reply from any RTU), it is better to check if the target RTU is busy as shown below.



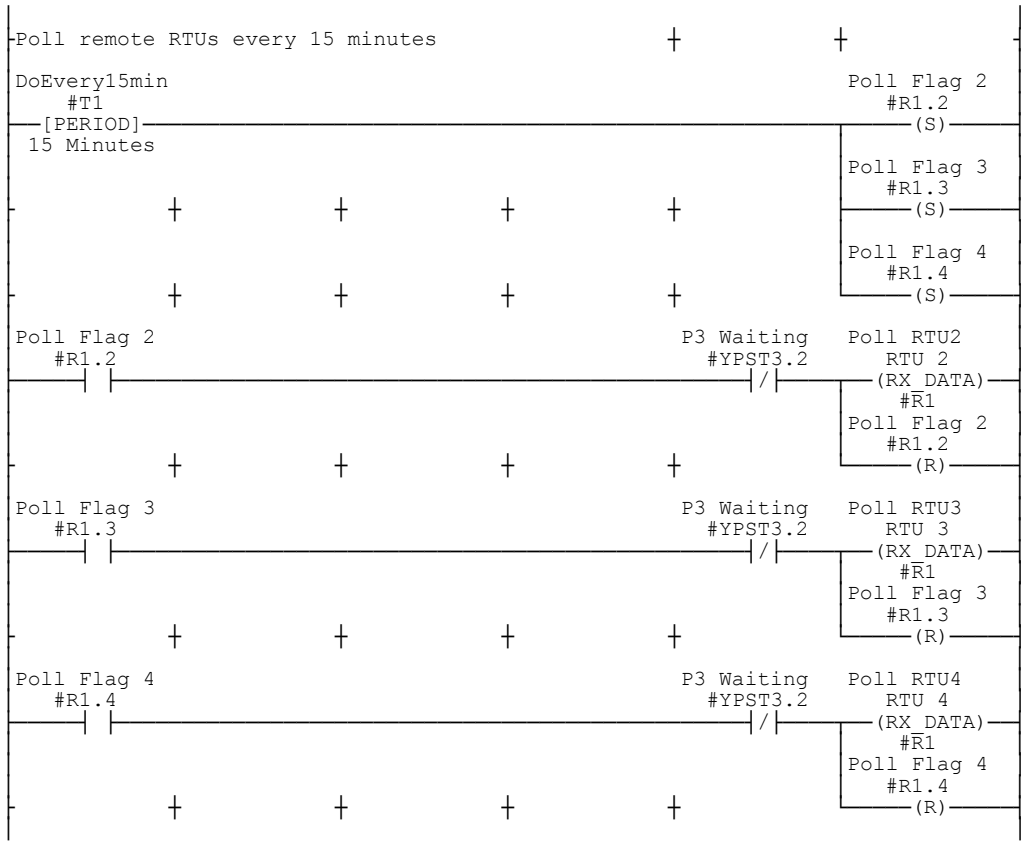
Example - Polling Data

Polling is usually performed by the master RTU in order to get a regular update of remote RTU data and to determine if communications to the remote RTUs have failed.

The Series 2 protocol allows for full-duplex communications which means that the RTU can simultaneously transmit and receive. However, since most radios are half-duplex, which means that the RTU cannot transmit and receive simultaneously, it is necessary to force the RTU to wait for a reply to each transmit message. Unless the RTU is forced to wait, it will transmit all the polling messages one after the other - which can take a few seconds. Because of the delay, the first polling message will have timed out before the RTU is able to receive a reply as the RTU is still busy transmitting. The following examples show how to force the RTU to wait for a reply to each message when polling.

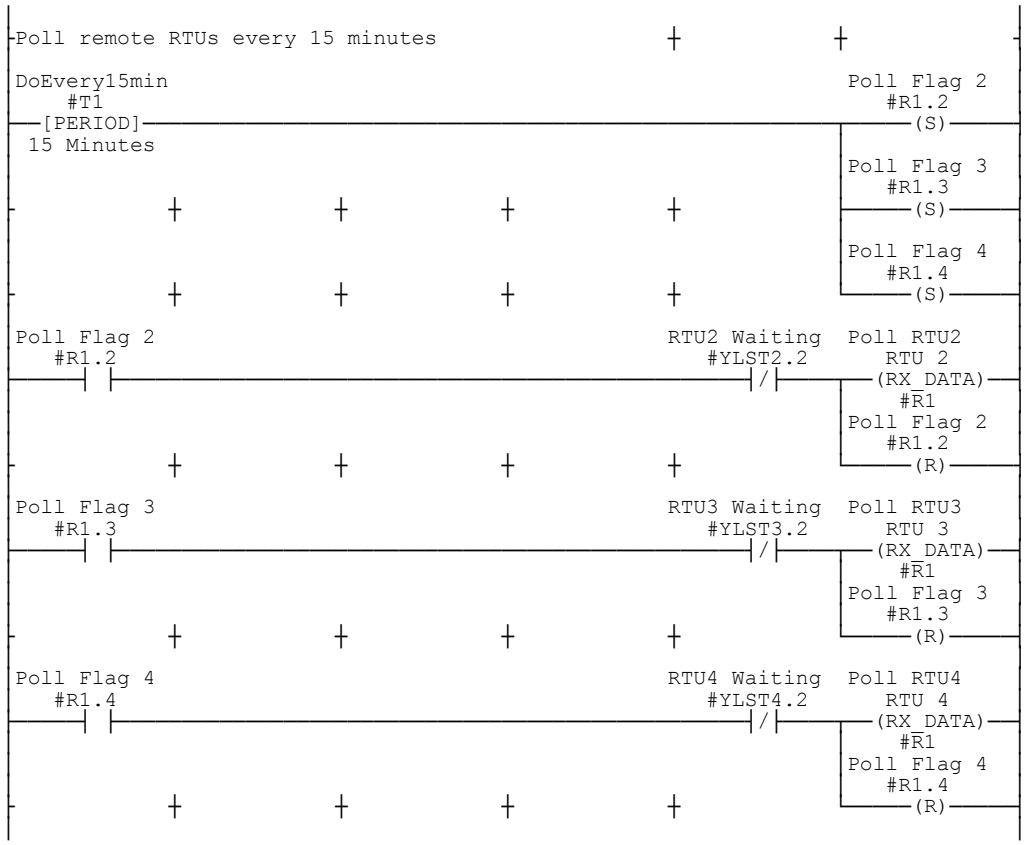
Basic polling

The example below polls RTUs 2, 3 and 4 every 15 minutes. The example checks that port 3 is free (ie. not waiting for a reply to a message - #YPST3.2) before polling the next RTU. Note: the port number to check is the port configured to communicate with each remote RTU as defined in the Port List.



Basic Polling Using Ethernet

It is possible to communicate with multiple RTUs at the same time using one Ethernet port. Therefore, instead of checking if the port is busy (waiting for a reply from any RTU), it is better to check if the target RTU is busy as shown below.



Polling After Data Has Expired

If an outstation RTU has exception reported to the master RTU recently, it is not necessary to poll the outstation RTU until the data is older than X minutes (where X is ideally a SCADA setpoint with a default value of say 30 minutes). If exception reports are generated frequently, it may never be necessary to poll the outstation RTU. Communication statistics are still accumulated as a success is recorded for each exception report received. Communication Fails are also recorded as the master will still effectively check comms every X minutes if it has not heard from the outstation RTU. Note: the registers that are exception reported should be the same as the registers that are polled. If a Maximum Quiet Time setpoint is used, this should be loaded with a default value on the first ladder scan.

Advantages:

- Minimises communications over the network.
- User can set the maximum age of data before a poll.

Disadvantages:

- Takes more effort to test.

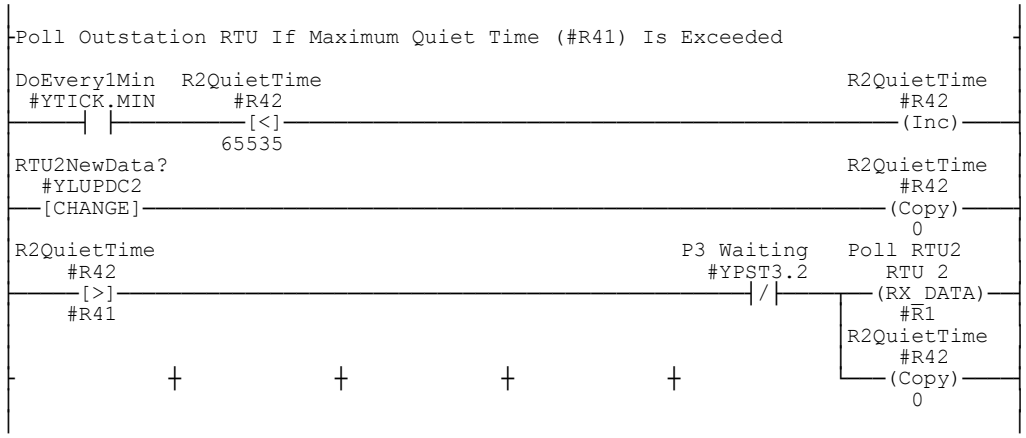


Figure: Polling An RTU After No Exception Reports For X Minutes. Note: Replace *P3 Waiting #YPST3.2* with *RTU2 Waiting #YLST2.2* if using an Ethernet port.

Example - Event Logging

Event Log Pointers
Event logs are kept in a circular buffer that is Max number of logs long (as defined in the topic Configuration - Memory). When the buffer is full, the oldest logs are overwritten. The RTU uses an internal *current pointer* to point to the latest log added to the buffer.

If an outstation RTU sends its event logs to more than one master RTU, the outstation should use a different pointer for each master RTU. This allows the initiating RTU to know how many event logs it has sent to each RTU.

Moving Event Logs Through An RTU Network
Event logs can be transmitted over an RTU network and accumulated in the master RTU. The event logs from all the outstations are stored in the one event log buffer in the master RTU.

It is not advisable to have the master RTU requesting logs *and* the outstations sending logs in the one system. This is because the event log pointers in the master RTU and in each outstation RTU will not be synchronized and the event logs already sent by the outstation will be requested again by the master and vice versa. This results in two identical event logs in the master for each outstation event log. Therefore event logs should only be moved in one direction (ie. either the outstations can send the event logs OR the master can poll the event logs).

Generating Event Logs
Before generating event logs, some memory must first be allocated for storing the logs (please see the topic Configuration, Memory, Event Logs. Eg. 32kB of memory can store 2730 logs). Event logs can then be generated using ladder logic as shown below.

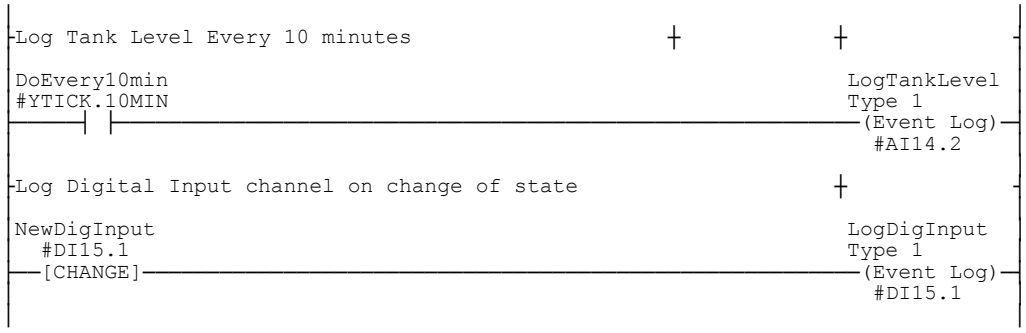


Figure: Generating Event Logs

Example - Polling Event Logs

An Rx Update block can be used to poll event logs and data from up to 16 RTUs. The system parameters - *Update Register Blocks* and *Update Hardware Blocks* are configured in the outstation RTU to control which of the data blocks will be checked or read from the RTU (please see the topic Configuration - System Parameters for more information). The RX Update block can also poll all the event logs that match the configured priority and user type until the maximum limit is reached or until there are no more event logs. In order to use the Rx Update block, the RTU requires the RX Update driver - *RXUPDxx.Dxx* to be loaded.

In the example shown below, the RTU checks if the Rx Update block is finished (ie. all the pending flags are set off) before initiating a new Rx Update. This avoids initiating multiple RX Update messages which could cause unpredictable results (event logs may be lost or overwritten). A single Rx Update block can generate many messages causing the port pending bit to be set and reset many times. New message blocks should not be initiated from the same port until the Rx Update block is completely finished.

#R52 is used for the *Pending Flags* register in the Rx Update block. On the first ladder scan, #R52 is initialised to zero as #R52 is not reset to zero after a warm start. This clears any pending flags that may still be set if the RTU was stopped in the middle of an RX Update.

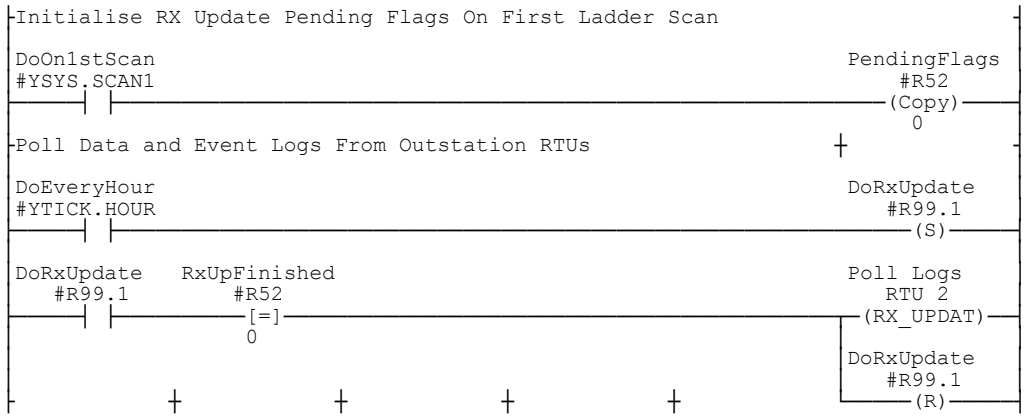


Figure: Example RX Update Ladder To poll multiple RTUs.

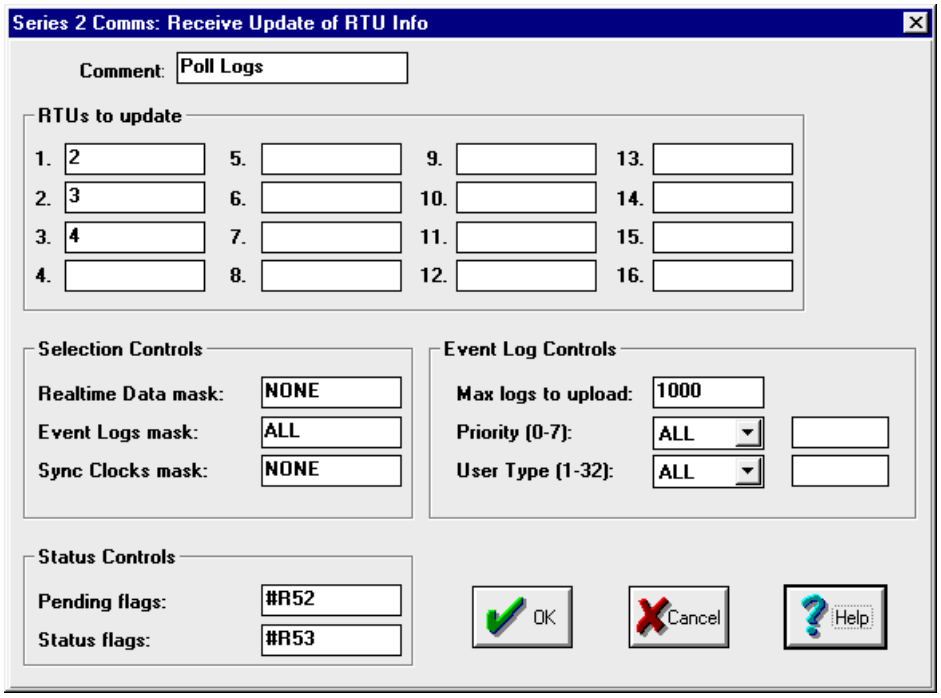


Figure: Rx Update block used above to poll event logs from RTUs 2, 3, 4.

Example - Comms Fails Today And Yesterday

Communication fails give a good indication of the state and reliability of the RTU communications network. Communication statistics are automatically recorded by the RTU in Network Link Registers. These network link registers can be accessed by SCADA software by copying them to local registers. At midnight, the fails today values can be copied to fails yesterday registers and then the network link registers reset to zero. Communication status is updated after each poll or when an exception report is received (which clears the comms fail).

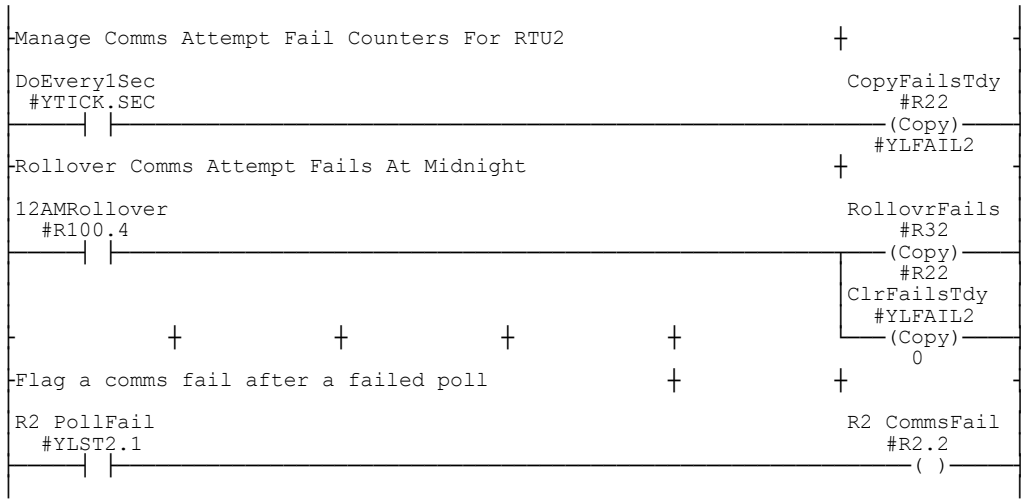


Figure: Managing Comms Attempt Fails For Today And Yesterday

It is usually more accurate to record the number of message fails instead of the number of attempt fails. A message fail occurs when all of the configured attempts fail (eg. 3 attempts at each poll message). The example below shows how to count message fails and how to flag a communications fail after 10 consecutive attempt fails for RTU2.

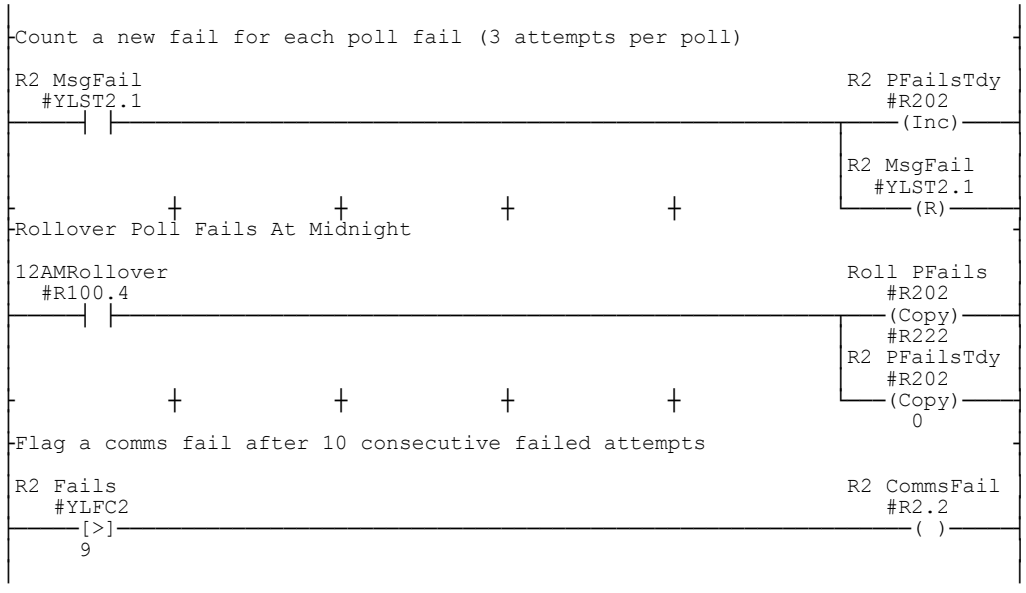


Figure : Managing RTU2 Poll Fails For Today And Yesterday

Example - Modems (PSTN, GSM and 3G)

PSTN, GSM and 3G (Maxon Modmax MM-6280IND) modems can be used on any serial port of the RTU by configuring the following items:

- **Configuration, Port List:** Set the port *Type* to PSTN and then select the **Configure** button. Set the *Init String* for the type of modem being used. If the PSTN modem or GSM is to be used for dialling a paging service, the default initialisation string may need to be changed as detailed in the topic Configuration - Port List, PSTN, Init String.
  When using a 3G modem, a blank initialisation string should be used. The following settings should be set in the 3G modem before using it with the RTU. Connect the 3G modem to a PC using the cable supplied with the modem (or by using an RJC-ADP-26 cable with an ADP-08 adapter from Semaphore). Use the Toolbox terminal program (or Windows HyperTerminal) initially at 115200 baud (the default modem baud rate) to set the following AT commands. Note the data format is 8 data bits, 1 stop bit, no parity bit, no flow control.

AT+IPR=9600	Sets baudrate to 9600 between the modem and the RTU (after this AT command, Toolbox will need to be set to 9600 baud)
AT&C1	Ensure carrier detect is not always on
ATS0=2	Answer after 2 rings
ATV0	Disable verbose (text) responses
ATE0	Disable Echo mode (the AT commands will not be displayed on the screen after this)
AT&W1	Save above settings in user profile 1
AT&F1	Load above settings into the active profile

- **Configuration, Port List:** For 3G modems, set *Baud Rate* to 9600. When using a PSTN or GSM modem to dial a paging service, 9600 can also be used. For PSTN and GSM modems, leave *Pre TX* and *Post TX* set to 0. For 3G modems, set *Post TX* to 400 ms (this allows for breaks in messages received from the 3G Network) and leave *Pre TX* as 0.
- **Configuration, Network List:** Set *Target RTU* to the address of the RTU to dial and *Port #* to the port number configured above. The default *Timeout* of 2000ms can be used in most cases. Note: a network link is not required if the RTU is only answering calls.
- **Configuration, Phone Directory:** Set *Primary Phone Number* and *Secondary Phone Number* to the phone number of the target RTU to dial. Note: the phone directory does not have to be configured if the RTU is only answering calls
- Configure a communications block (eg. RX_DATA) in ladder logic to communicate with the target RTU. The RTU will then automatically dial the number configured above.

Connecting a GSM: In order to dial *into* a GSM, a data telephone number is required. This is a second telephone number. To obtain a Telstra data number in Australia, call Telstra on 1800 730 053. When a GSM SIM card is obtained, please ensure that it does not require a PIN number (a PIN number is not supported by a GSM) and also ensure that it is a non-transparent data number. If the GSM unit is only used for dialling *out* from the RTU, a data number is not required and the normal voice number can be used. The SIM card can be checked that it is enabled on the network by installing it in a mobile phone.

Connecting a 3G Modem: [When used as a dialup modem] In order to dial *into* a 3G modem, a data telephone number is required. This is a second telephone number. To obtain a Telstra data number in Australia, call Telstra on 1800 730 053. When ordering a 3G SIM card, please ensure that it has **data package bearer code** of 2620 (this ensures that the data number is **non-transparent**) and request that Voicemail and Missed Call Notifications be disabled. Ensure that the SIM PIN number is disabled. If the 3G modem is only used for dialling *out* from the RTU, a data number is not required and the normal voice number can be used.

Testing a 3G Modem: The SIM card can be checked that it is enabled on the network by installing it in a 3G mobile phone. A 3G modem can be tested by dialling it from a standard PSTN telephone line. If successful, the 3G modem will answer and attempt to establish a data connection. Note: dialling a 3G modem from a mobile or PABX phone will result in *unable to connect* (from a mobile) or an *engaged* signal (from a PABX). This is because most mobile and PABX phones have header data that allow the mobile network to determine that it is a voice call and so will not allow a data connection.

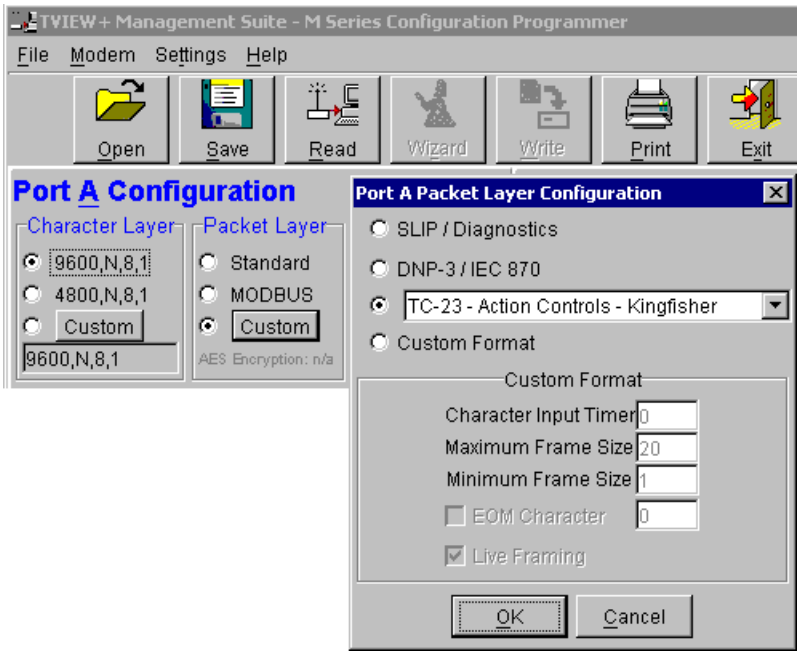
For diagnosing faults when using a modem, please see the appendix - RTU Commissioning, Trouble Shooting.

Example - Radios

Various types of external radios require various types of RTU ports and cables. The external radios that are commonly used by Semaphore and the RTU setup for each radio is detailed below. Note: the spread spectrum radio option board is detailed in the topic Driver - Spread Spectrum Radio.

Radio	RTU Option Board	Port Type	Port Baud Rate	Port Pre / Post TX (ms)	Network Link min. Timeout
Trio MR450 with no modem, Trio TC-450SR, TC-900SR, Maxon SD-125, Tait T2010	PC-1/MC-1 Radio	Radio	1200	300 / 100	2200
	Line-2	Line-2	1200	300 / 100	2200
Trio MR450 with 2400 / 4800 / 9600 modem *	Any Serial board	RS232 Radio	2400 / 4800 / 9600	0 / 0	2000
Trio TC-450SR with 24SR Trio TC-900SR with 24SR	Any Serial board	RS232 Radio	2400	0 / 100	2000
Trio TC-450SR with 48SR Trio TC-900SR with 48SR	Any Serial board	RS232 Radio	4800	0 / 100	2000
Trio TC-450DR Trio TC-900DR	Any Serial board	RS232 Radio	4800 / 9600	0 / 0	2000

* Trio M Series radios are configured using TVIEW+ Software. TVIEW+ allows the transmit and receive frequencies of the radio to be set. The **Packet Layer** setting can also be changed to **Custom** (instead of **Standard**). When Custom is selected, a window will appear allowing **TC-23 - Action Controls – Kingfisher** to be selected (as shown below). This will configure the Trio radio to store and forward complete Kingfisher messages.



Example - Satellite Phones

A satellite phone is treated like a PSTN modem by the RTU and can be used on any serial port. The following setup is for a Motorola 9522 satellite phone.

- First ensure that the latest firmware supporting Satellite Phones is loaded in the RTU (as detailed in **protocols.pdf** available from http://helpdesk.servelec-semaphore.com).
- When a satellite phone is first obtained, it usually requires a PIN number to boot up (if this has not already been disabled). Use AT+CPIN? to check if it needs a PIN. To set the PIN number use AT+CPIN="1111"<CR> (1111 is the default PIN number). The command AT+CLCK="SC",0,"1111"<CR> should then be used to disable the need for a PIN number.
- The satellite phone antenna *must* have a very clear view of the sky and not be near any buildings or objects (it should have a 150 degree uninterrupted view of the sky). Use AT+CSQ to query signal strength. This will return a number in the range 0-5 where 3-5 is good.
- **Configuration, Port List:** Set the port *Type* to PSTN, *Pre Tx* to **20,000** ms. Select the **Configure** button to set the initialisation string. Use the default string for an external modem (AT&FTE0V0S0=2&W). Note: a Pre TX delay of 20,000 ms *must* be used as the satellite phone uses the off hook signal to set carrier detect. The satellite phone signals that a carrier is detected and that it is online before the handshaking has occurred with the remote modem. Configuring a Post Tx delay of 20,000ms forces the RTU to wait for 20 seconds before attempting to send a message.
- **Configuration, Network List:** Set Target RTU to the address of the RTU to dial and *Port #* to the port number configured above. The default Timeout of 2000 ms can be used. Note: a network link is not required if the RTU is only answering calls.
- **Configuration, Phone Directory:** Set *Primary Phone Number* and *Secondary Phone Number* to the phone number of the target RTU to dial. Note: the phone directory does not have to be configured if the RTU is only answering calls. The international dialling code for a satellite phone to dial out is 00 <country code><area code><phone number>. The prefix *00* is always needed by a satellite phone when dialling a phone number. Eg. to dial Semaphore (Australia) use 00 61 3 8544 8544 where 61 is the country code and 3 is the area code.
- Configure a communications block (eg. RX_DATA) in ladder logic to communicate with the target RTU. The RTU will then automatically dial the number configured above.

SMS Pager Messages: A satellite phone can send pager messages to the Telstra PET SMS service. In the Pager Configuration window, set Phone No. to 00 61 439 125107 (the international Telstra SMS Access Manager phone number). The satellite phone can currently only send SMS messages *directly* to other satellite phones.

Dialling A Satellite Phone: Each satellite phone has a data number and a voice number. An RTU uses the data number. To dial the satellite phone from Australia use 0011 <satellite phone data number>. Eg. 0011 8816 123 45678 where 0011 is the international dialling code when in Australia and 8816 123 45678 is the satellite phone data number.

Communicating With A GSM: Dialling a GSM modem using a satellite phone is unreliable and not recommended.

Example - GPRS Modems

GPRS (General Packet Radio Services) modems maintain a continuous connection to the GSM mobile network and provide faster data transmission rates than GSM modems. However, data transmissions are not continuous. Data is broken into *packets*, allowing multiple GPRS modems to share the same channel. This can cause some data transfers to take up to 10 seconds. The following setup is for a Wavecom Fastrack GPRS modem (with TCP/IP stacking enabled) connected to an RTU serial port.

- A Wavecom Fastrack GPRS modem requires a few AT commands to get it to work properly. After connecting the GPRS modem to the PC (an RJC-ADP-22 cable and an ADP-08 adapter from Semaphore can be used), use Toolbox terminal at 115200 baud (the default Wavecom Fastrack baud rate) to set the following AT commands:

AT#APNSERV= "xxxxx.corp"	Sets the Access Point Server Name (xxxxx.corp) as obtained from the service provider.
AT#APNUN="user@ xxxxx.tpips.com.au"	Sets the Access Point User name (user@xxxxx.tpips. com.au) as obtained from the service provider.
AT#APNPW="zzzzz"	Sets the Access Point Password (zzzzz) as obtained from the service provider.
AT+DIALN1="*99***1#"	Sets the Dial Number. Note: this is different for different service providers.

- To verify the GPRS modem is setup and functioning correctly, the following AT commands can be used:

AT#VALL	Lists all of the details, including IP details of the modem.
AT+CGMR	Displays the firmware version of the modem. For version "641_09gg.Q2406B 1328940 111903 18:23" use the **GPRS** RTU driver. For version “655_09gg.Q2406E 2015268 111705 17:01” or newer use the **GPRS2** RTU driver.
AT#VVERSION	Displays the software version of the TCP/IP Stack. Recommended version is "eDsoft_W302_V2.10 1166 86 Dec 10 2003 12:20:17" (or newer).
AT#VGPRS	Displays the configured server name, user name and password.
AT+CSQ	Displays the signal strength. Returns 0 to 31 or 99 (no signal). The minimum value for successful communications is 15 or higher.

- Ensure that the GPRS driver is supported by the type of RTU that is being used and that the latest firmware and driver are loaded in the RTU (as detailed in **protocols.pdf** available from http://helpdesk.servelec-semaphore.com).
- **Configuration, Port List:** Set the port *Type* to GPRS or GPRS2 (depending on which firmware version is in the GPRS modem as detailed above). Set *Baud Rate* to 115200 and *Post Tx* to 1500 ms. Select the **Configure** button to set the GPRS parameters. Set *Dial Retries* to 2 and *Dial Timeout, Online Inactivity* and *Hang Up After* to 60 seconds.
  Note 1: Toolbox cannot communicate with a port configured as GPRS. At least one port (preferably port 1) must be configured for Kingfisher Series 2 protocol. To restore Toolbox communications on a GPRS port, remove the CPU RAM battery link to clear the configuration. Note 2: *Online Inactivity* and *Hang Up After* timeouts should not be set to 0 seconds as this will cause the GPRS modem to stay online indefinitely. If the local GPRS modem disconnects while the remote GPRS modem is configured to stay online indefinitely, it will not be possible to communicate with the remote GPRS modem until it is reset.
- **Configuration, Network List:** Set *Target RTU* to the address of the RTU to contact and *Port #* to the port number configured above. Set *Timeout* to 6000ms (or greater). Set *IP Address* to the IP address of the GPRS modem of the Target RTU. Note: a network link is not required if the RTU is only receiving calls.
- Configure a communications block (eg. RX_DATA) in ladder logic to communicate with the target RTU.
- Additional GPRS information can be obtained from the port registers #YPSIGnn (signal strength), #YPSTnn.6 (online status) and #YPERRnn (initialisation status).

Connecting a GPRS Modem

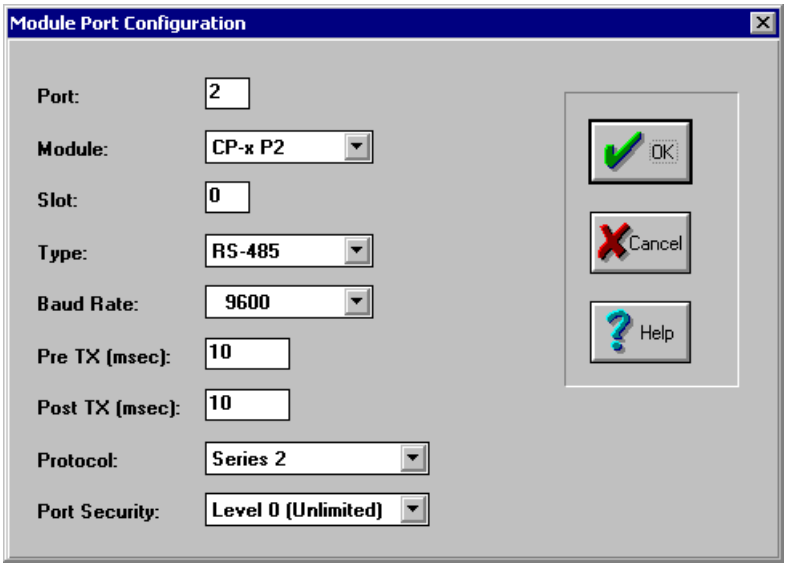
It is very important to request *fixed IP addresses* for all GPRS modems. A SIM card and the following information will also be required:

- Access Point Server Name - xxxxx.corp where xxxxx could be Telstra or the company name eg. Acme.corp
- Access Point User Name - user@xxxxx.tpips.com.au where *xxxxx* is the company name and *user* is the name of the GPRS modem.
- Access Point Password
- For more information please see the document *GPRS Version Xxx.PDF* available from Semaphore.

Example - RS485

RS485 can be used on any CP-xx *isolated* serial port or PC-1 serial port 2.

- **Configuration, Port List:** Set *Type* to RS485 and set *Baudrate* to match the remote device (eg. 9600). Set *Pre Tx* and *Post Tx* to 10 ms as illustrated below (note: Post Tx can be set to 1ms for fast response RS485 devices).



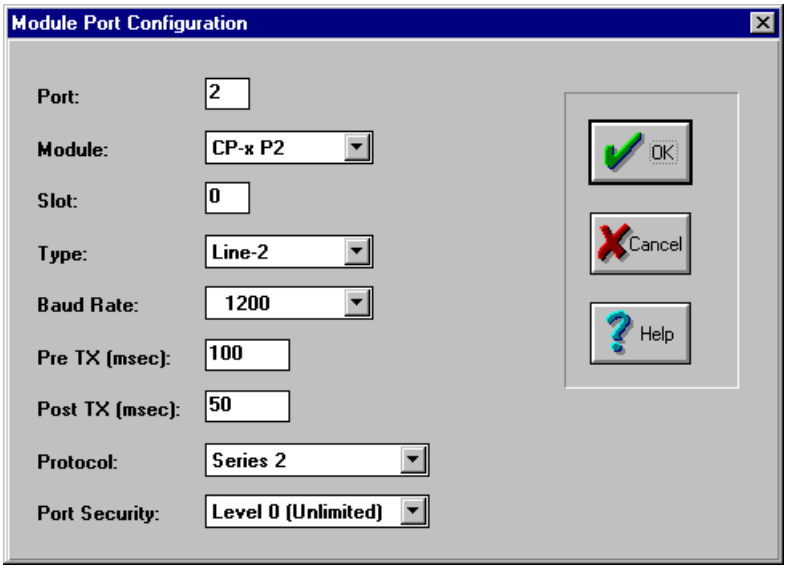
- **Configuration, Network List:** Set *Target RTU* to the address of the RTU to communicate with and *Port #* to the port number configured above. Note: a network link is not required if the RTU is not initiating messages.
- Configure a communications block (eg. RX_DATA) in ladder logic to communicate with the target RTU.

Note 1: The length of the RS485 cable should be at least 1m.
Note 2: Each end of the RS485 cable should be terminated with a 120 ohm resistor. For short cable runs (< 5 m) a single terminating resistor of 60 ohms can be used at one end of the RS485 cable. Please see the Kingfisher Hardware manual, CP-xx Serial Option Board wiring diagram for details.
Note 3: If problems persist, add a Pre TX delay to the remote device or decrease the baudrate (do not use less than 4800 baud).

Example - Private Line

Two or four-wire Private Line communications can be used on any CP-xx LINE-L or LINE-2 port. Two-wire private line communications can also be used on PC-1/MC-1 Line option ports.

- **Configuration, Port List:** Set *Type* to LINE-2 (for LINE-2 ports) or PLINE (for LINE-L or PC-1/MC-1 Line option ports). Set *Baud Rate* to 1200, *Pre TX* to 100 (ms) and *Post TX* to 50 (ms). Note: Pre and Post TX can be reduced to 50 and 10 ms respectively in some installations.



- **Configuration, Network List:** Set *Target RTU* to the address of the RTU to communicate with and *Port #* to the port number configured above. Note: a network link is not required if the RTU is not initiating messages.
- Configure a communications block (eg. RX_DATA) in ladder logic to communicate with the target RTU.

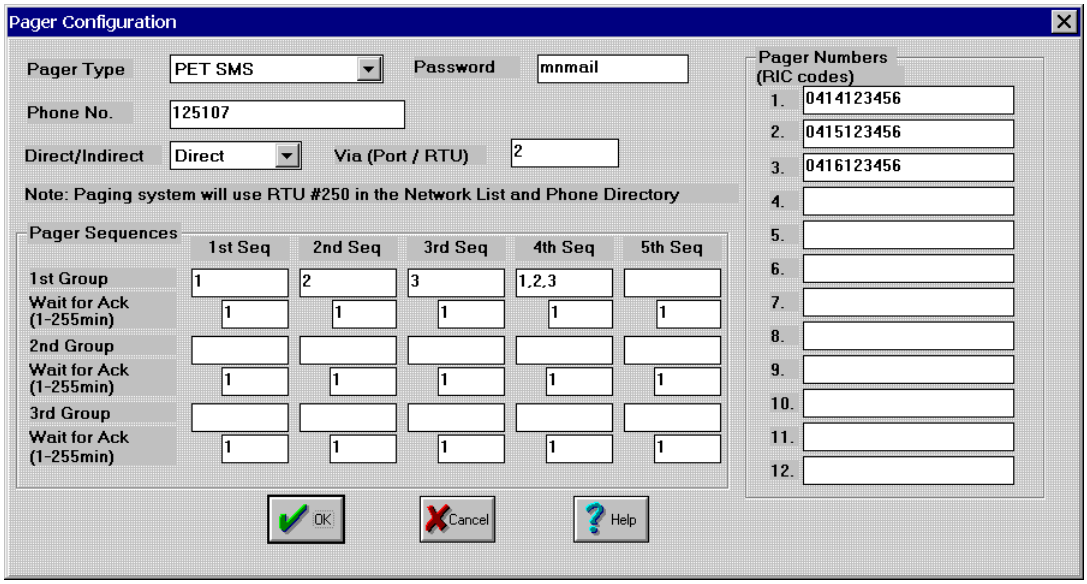
Note: Each end of the private line cable should be terminated with a 600 ohm resistor. Please see the Kingfisher Hardware manual, CP-xx Line/Radio Option Board for wiring details.

Example - SMS Pager Messages

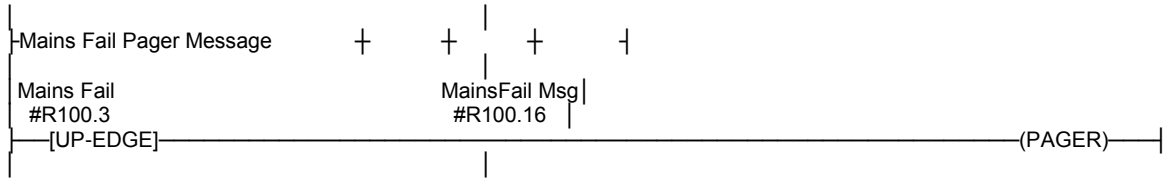
The following example uses a PSTN modem to dial Telstra SMS paging in Australia and send an SMS message to one mobile phone. The RTU has a PSTN modem on port 2 of the CPU. The phone numbers of the mobile phones are 0414 123456, 0415 123456 and 0416 123456. The phone number of the Telstra PET SMS paging service is 125 107. Note that 125 107 only provides service to Telstra lines (if your account is with Optus etc. it won't work).

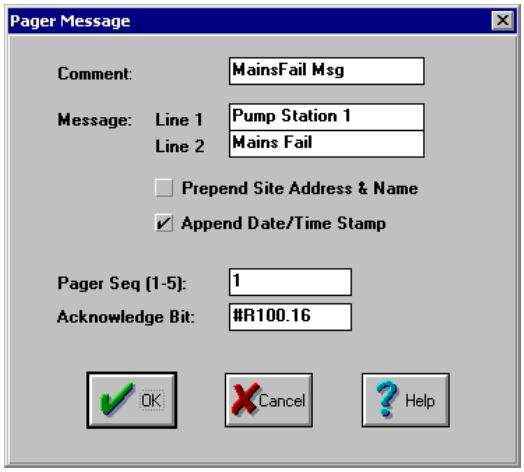
The figure below shows how these paging parameters have been configured from *Configuration - Pager Configuration*. Note: please see the topic Configuration - PSTN for details on setting up modem initialisation strings for dialling a paging service.

Note: the password below will allow one message to be sent to *one* Pager Number each time the paging service is dialled. Telstra also supports sending one message to multiple Pager Numbers if an individual password is obtained. To obtain a password, call Telstra on 1800 730 053 and ask about SMS Access Manager.



The ladder below shows how a register bit is used to trigger a pager message.





Pager message block used in the ladder logic above

Since the pager message is only transmitted once, the Acknowledge Bit (#R100.16) is not used. If the pager message was to be transmitted more than once (by configuring a 1 in the *2nd Group* and *3rd Group* of the 1st Sequence), writing a 0 to #R100.16 would acknowledge the pager message and stop the pager message from being re-transmitted.

GSM SMS Pager Messages

A GSM can be used on any CPU serial port (PC-1, CP-1, CP-10/11/12/21, LP-1/2/3) to send an SMS message to any mobile phone (Optus, Vodaphone, Telstra etc). When using a GSM instead of a PSTN modem, configure *Pager Type* as GSM SMS and leave *Phone No.* and *Password* blank (these settings are not used). A GSM allows a single pager message to be sent to multiple phones when a sequence with multiple phone numbers is used (eg. as shown in the 4th Sequence above).

Dial-Up SMS Pager Messages - Sending To Multiple Mobile Phones

After obtaining a password, Telstra's SMS Access Manager allows each pager message to be sent to multiple pager numbers (this would occur if a pager message block was configured with Pager Sequence 4 as shown above). If the paging service only allows one message to be sent to one phone number at a time (eg. if using the Telstra paging service and the *mnmail* password), a message can be sent to multiple phones by:

- Configure 1 phone number in each pager sequence (eg. as shown in pager sequences 1 to 3 above)
- Configure ladder logic that triggers 1 pager message block for each phone number when an alarm occurs. Each pager message block should be configured to use a different sequence number that corresponds to the phone number.

The example below shows how each time the Mains Fail alarm occurs, 3 pager messages are triggered that target different phone numbers.

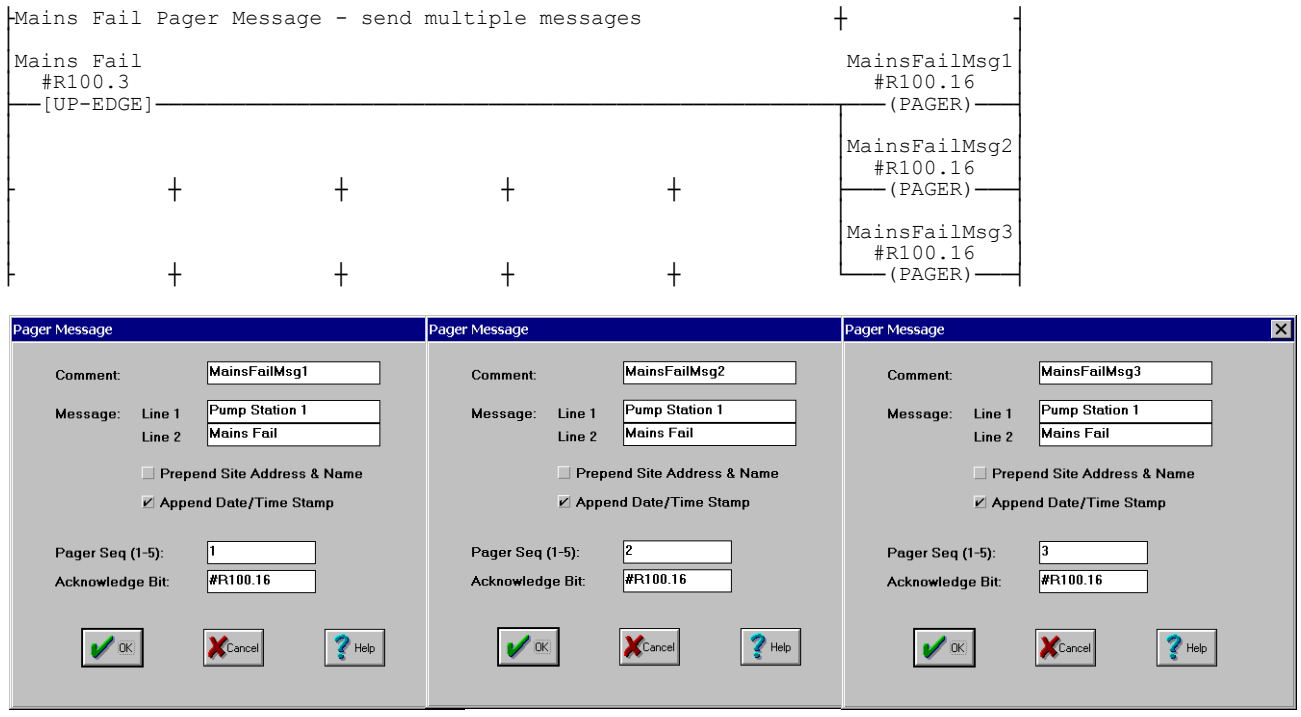


Figure: Ladder Logic and Pager Message Blocks Used To Send A Pager Message To 3 Phones

Telemetry System Paging

It is best to configure all the pager messages in the master RTU for two reasons:

- The amount of communications in the telemetry system is minimised.
- To enable or disable the pager receivers that messages are sent to, the bits in only one group register need to be changed instead of changing the group register in every RTU that initiates pager messages.

New data is usually exception reported to the master RTU and so this new data can be used to trigger a pager message.

If say #Rxx is used for all three groups in the first sequence configured in *Configuration, Pager Configuration*, then SCADA software can be used to set and reset the individual bits of #Rxx corresponding to the pager receivers to enable or disable. These bits could also be cleared during certain times to prevent pager messages at inconvenient times.

Example - Pager Messages With Variables

The previous example showed how a fixed pager message could be sent to a mobile phone as an SMS. Pager messages can also be sent that contain register values. This is achieved by sending a string of up to 31 ASCII characters. The string is overwritten with register values prior to sending the pager message. The string is stored in local registers with two characters in each register (8 bits are used for each character).

A string can be created using a String Copy block which creates a string up to 31 characters long. The String Copy then terminates the string with a null character (00 Hex).

To transmit a string in a pager message, *Line 1* of the pager message block is configured as the local register (#R1 to #R2048) that contains the first character of the string (in bits 1-8).

The example shown below converts a tank level stored as 0-999 in #R2 into ASCII characters and then transmits the tank level as a string in a pager message whenever #DI14.1 (Tank overflow alarm) is triggered. 0-999 is converted into ASCII characters by separating the number of 100s, 10s and 1s. Each digit is then converted into ASCII by adding 30 Hex (represented by 16#30 in ladder logic) to each digit. A standard string is created using the String Copy block containing *xxxxxxxxxx_Tank_Level_nn.n_%* and then *nn.n* is overwritten by the tank level ASCII characters. When overwriting the string with the tank level characters, care must be taken to overwrite the correct character positions. In this example the tank level is stored in characters 23-26 of the string. Since the string starts at #R401 (as configured in the String Copy block) and there are two characters in each register, the tank level is then stored in #R412 and #R413 as illustrated below. The first character in each register (ie. the left character) is stored in bits 1-8 (LSB). The second character is stored in bits 9-16 (MSB). Note: the LP-1/2/3 stores the left character in the MSB and the right character in the LSB.

Register #	401	402	403	404	405	406	407	408	409	410	411	412	413	414	415
Characters	xx	xx	xx	xx	xx	_T	an	k_	Le	ve	l_	nn	.n	_%	null

Note: letters denoting the site name can be used instead of the x's at the start of the string.

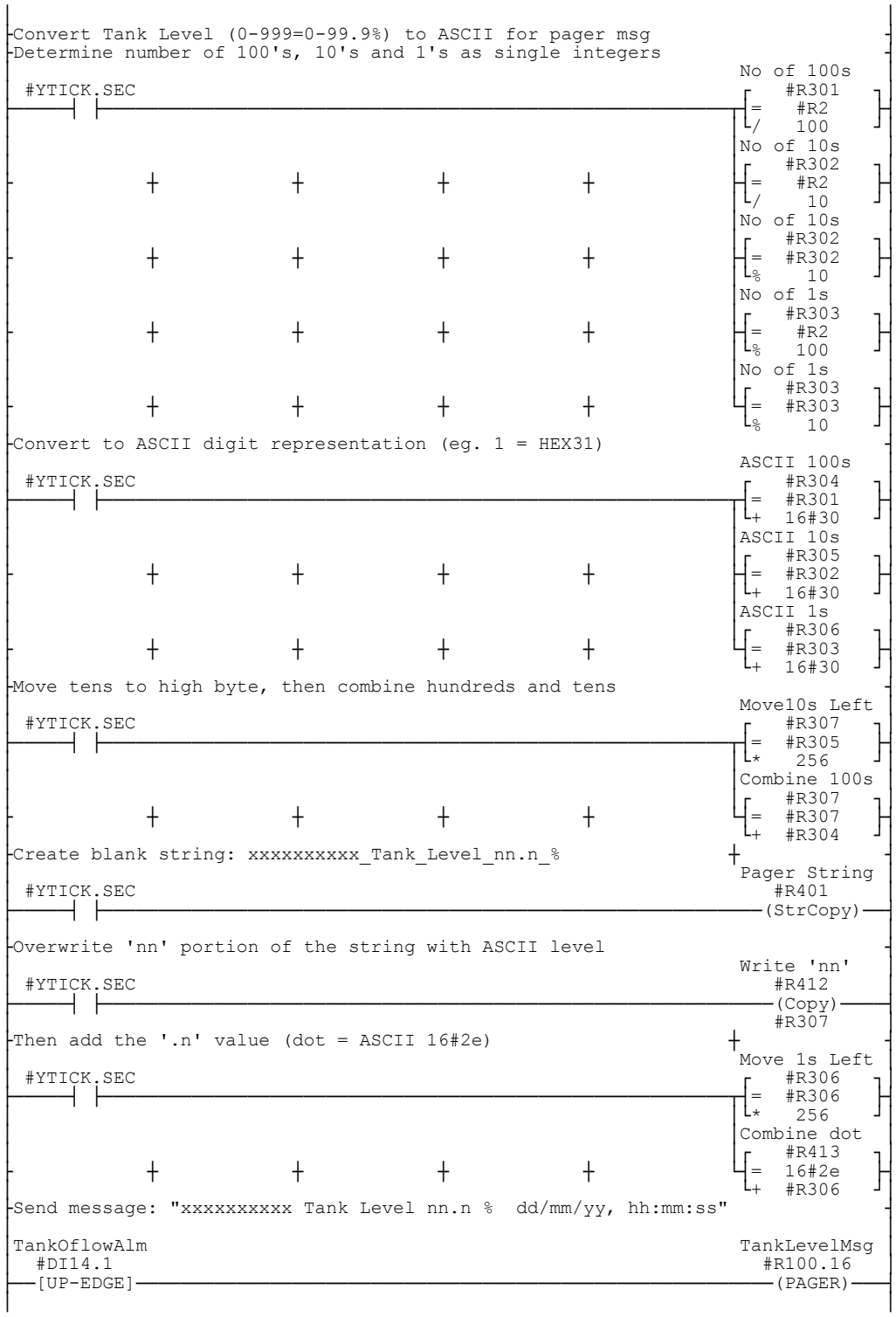


Figure: Generating A Pager Message Containing A Variable *nn.n*

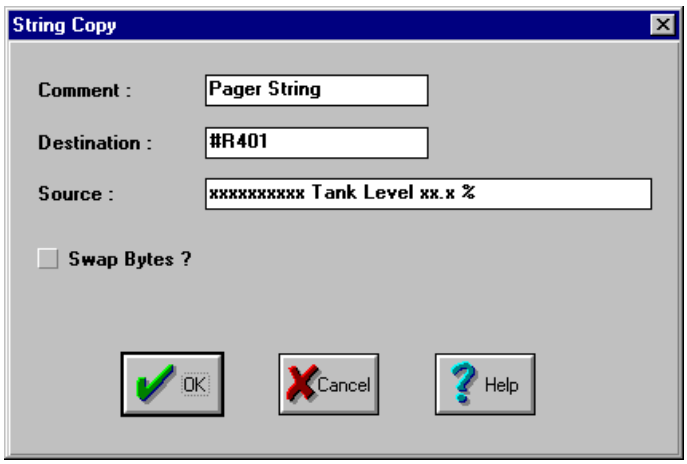


Figure: String Copy block used to make the basic pager message string

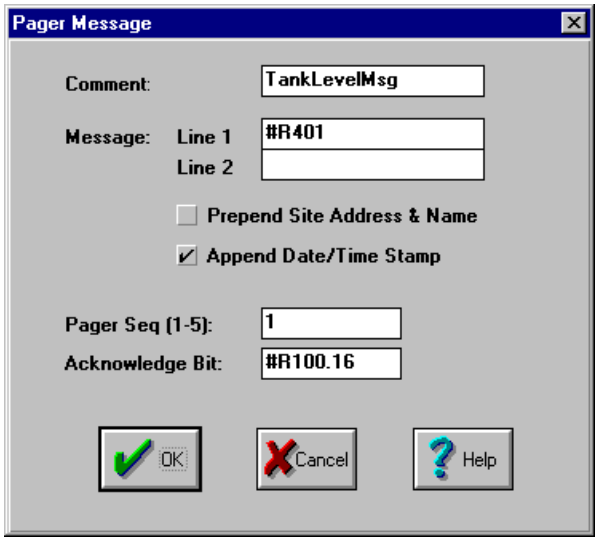


Figure: Pager message block used to transmit a string containing a variable.

Example - Kingfisher Images

To configure an RTU to capture images:

- First ensure that the latest firmware is loaded in the CPU or MC module (as detailed in **protocols.pdf** available from http://helpdesk.servelec-semaphore.com). The CPU firmware version can be checked by viewing the RTU Status and the MC firmware version can be checked by viewing the Hardware Overview and selecting the MC module.
- If a PC-1/CP-1 is being used with an MC-xx, then the image capture driver (IMAGExx.DRV) will need to be downloaded into the PC-1 or CP-1 using the menu **Utilities, Advanced, Download Firmware Driver**. When this is downloaded, "Image" will appear in the "Firmware drivers included" section of the RTU Status.
- Configure enough memory in the RTU for image storage using the menu **Configuration, Memory,** Image Buffer. A good quality large image uses 10 KB of memory. It is recommended that enough memory is allocated to store at least 2 images. When the image buffer is full, the oldest image is over-written by the newest image. When using an MC module, 256kB of memory is automatically allocated for image storage in the MC module.
- Configure the image capture port using the menu **Configuration, Port List.** The image capture option board is treated like a communications port. Set the following options:
  Module = CP-x P2 or CP-x P3 or MC-x P2 or MC-x P3
  (ie. the location of the image capture option board)
  Type = Image Capture
  All other parameters are irrelevant and are ignored.
- After downloading the RTU configuration, images can now be captured using the Image Manager program. This program allows the image capture port to be configured, images to be captured and then uploaded from the RTU.
- Alternatively, the RTU can use ladder logic to configure the image capture port and then capture images (please see the example below).

An example of setting up a single camera on port 3 of a CP-xx module and capturing images when a digital input is triggered is shown below. Every scan, the image buffer statistics are copied to local registers. As soon as a new image is added to the memory buffer (ie. when the value of Next Image Number changes) or after 2 seconds of the camera not ready, the *CameraReady* flag is set and the RTU is now ready to capture the next image. Note: care must be taken when capturing new images. If a new image is captured before the last image has finished, the last image will be corrupted.

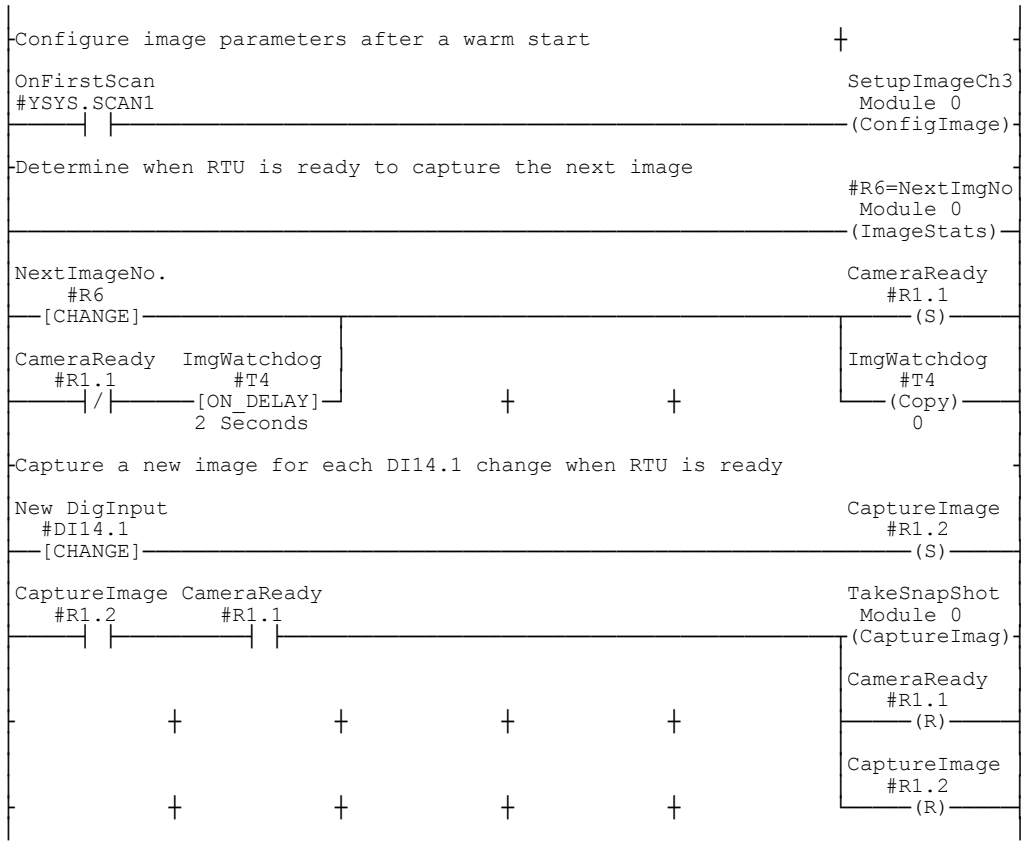


Figure: Capturing Images Using A Single Camera

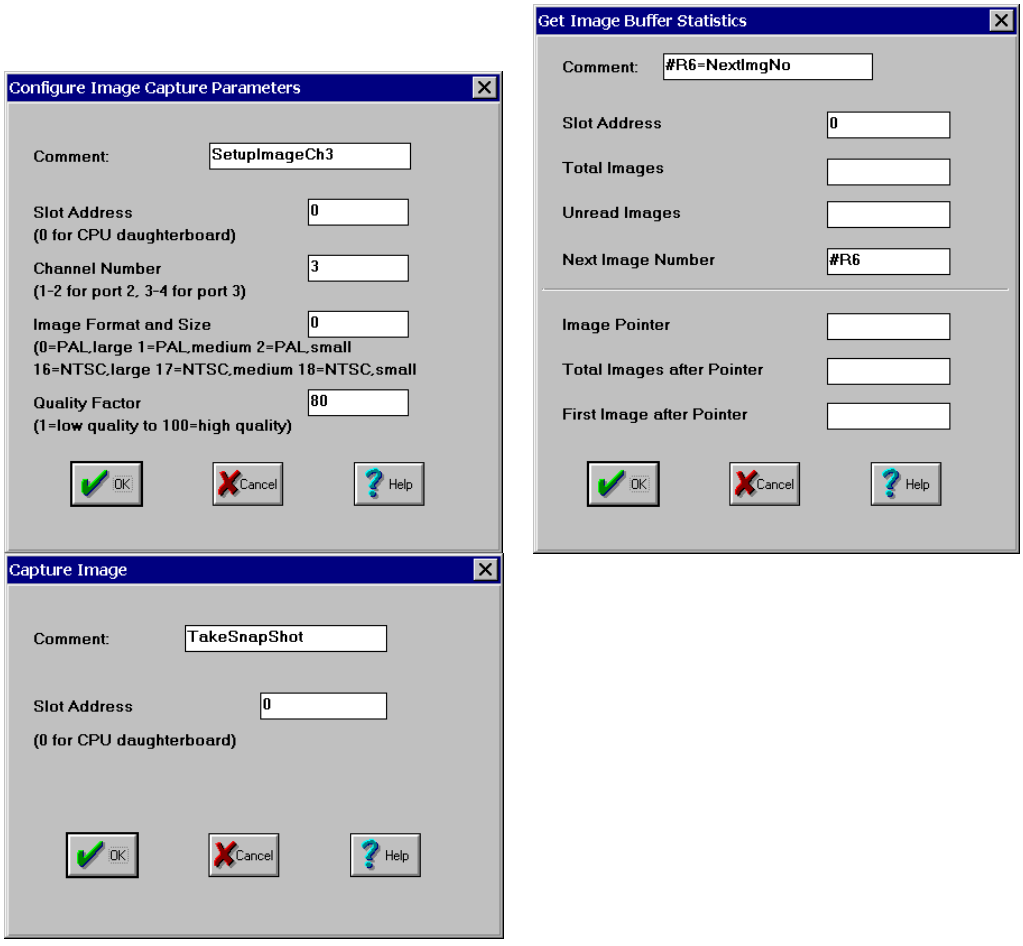


Figure: Image blocks used in the ladder logic above

Using Multiple Cameras

Each CP-10/11 or MC-10/11/12 module is treated as a 4 channel image capture module. One memory buffer is allocated for all the images from the 4 channels. Each MC-10/11/12 also has its own image buffer (256 KB). Before capturing an image, the image channel must first be configured (using the *Configure Image Capture Parameters* ladder block). It takes about 0.5 seconds for the image option board to be armed and ready to capture an image. Multiple images can then be captured from the one channel (these take about 0.5 to 1 second each to capture). Before using another camera channel, the new channel must first be configured and then after another delay of about 0.5 seconds, an image can be captured on the new channel. Note: each time the RTU is warm started or has its power reset, the image capture parameters need to be configured again.

Displaying RTU Images

RTU images can be uploaded and displayed on a PC using the Image Manager program. This program reads the images from a local or remote RTU, stores the image as a JPEG file and then displays the image on the PC. Images can also be deleted on the PC using Image Manager and deleted in the RTU by using the Warm Start command.

Trouble Shooting Image Capture Option Boards

When the CP-xx module is selected from the Hardware Overview (**View, Hardware Overview**) a window will appear, indicating which option boards have been detected on the CP-xx. The following settings can appear for the image board:

- “None” - the image board has not been detected. Check that the image board is installed correctly.
- "Image Capture" - the image board has been detected.

For details of the ladder logic image blocks, please see the topic Ladder Logic - Image Monitoring Functions.

Example - Low Power Mode

The RTU has two low power modes - *IO power saving* and *power down*.

IO power saving mode is configured from **Configuration, System Parameters,** I/O Power Saving Control. In IO power saving mode, the RTU can be configured to switch off various output voltages (eg. 24VDC from the BA-4 and from IO modules such as the AI-10, IO-4 etc) but in this mode the processor (eg. PC-1) keeps running.

Power down mode switches off the various RTU output voltages and also puts the processor to sleep. This is advantageous as the processor module uses more than 100mA at +5VDC. Putting the processor to sleep reduces the current consumption by over half. It is recommended that only one power saving mode be used - *IO power saving* or *power down*.

Power Down mode is controlled from ladder logic using two parameters #YPDTIME and #YPDSTAT (as detailed in the appendix RTU Data, System Registers). The RTU can be woken up in three ways: the power down time (#YPDTIME) expires, a cable with a CTS/RTS loop is connected to a configured serial port or a *Wakeup* message is received. To wake up the RTU using a cable with a CTS/RTS loop or using a Wakeup message, the RTU must be told which port(s) to monitor by setting the appropriate bit(s) in #YPDSTAT. An RTU may then be woken up with a Wakeup message if the initiating RTU has *Wakeup RTU From Power Down Mode?* checked in the network link for the powered-down RTU.

The example below shows how an RTU is put to sleep for 50 minutes at five minutes past the hour. The RTU will wake up after 50 minutes or if a serial cable with a CTS/RTS loop (eg. an RJ45 cable with an ADP-05 adapter) is connected to ports 1 or 2 or a Wakeup message is received on ports 1 or 2.

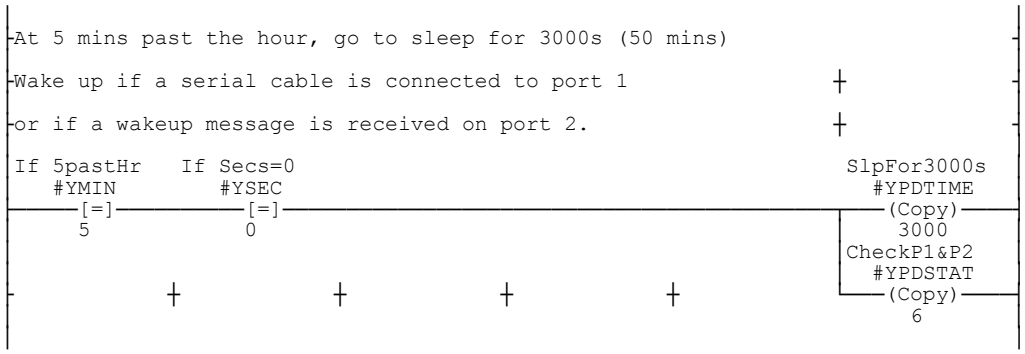


Figure: Power Down Ladder Logic

Example - RTU Diagnostics / Trouble Shooting

Sometimes it may be difficult to know why an RTU is behaving strangely. By logging the value of the RTU status register (#YSTAT) or another diagnostic register (eg. #YDIAG, #YFLAGS), the history of the RTU can be reviewed from the event logs.

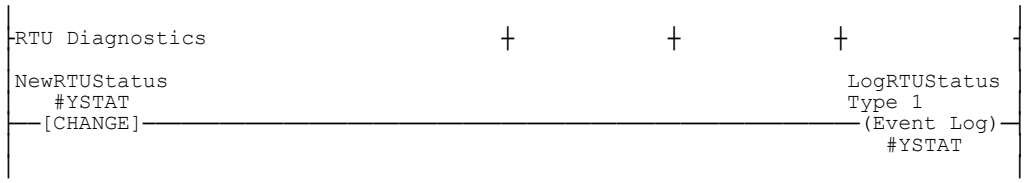


Figure: RTU Diagnostics by logging the RTU status register on any change.

Example - Indirect Addressing

This example shows how to add 10 consecutive local registers by using indirect addressing. Indirect addressing can reduce the amount of ladder logic required for repetitious tasks.

The following registers are used:
#R1 = pointer register (initial value is used to address the first of 10 consecutive registers to add)
#R2 = loop counter
#R3 = total
#R101 to #R110 = registers to add

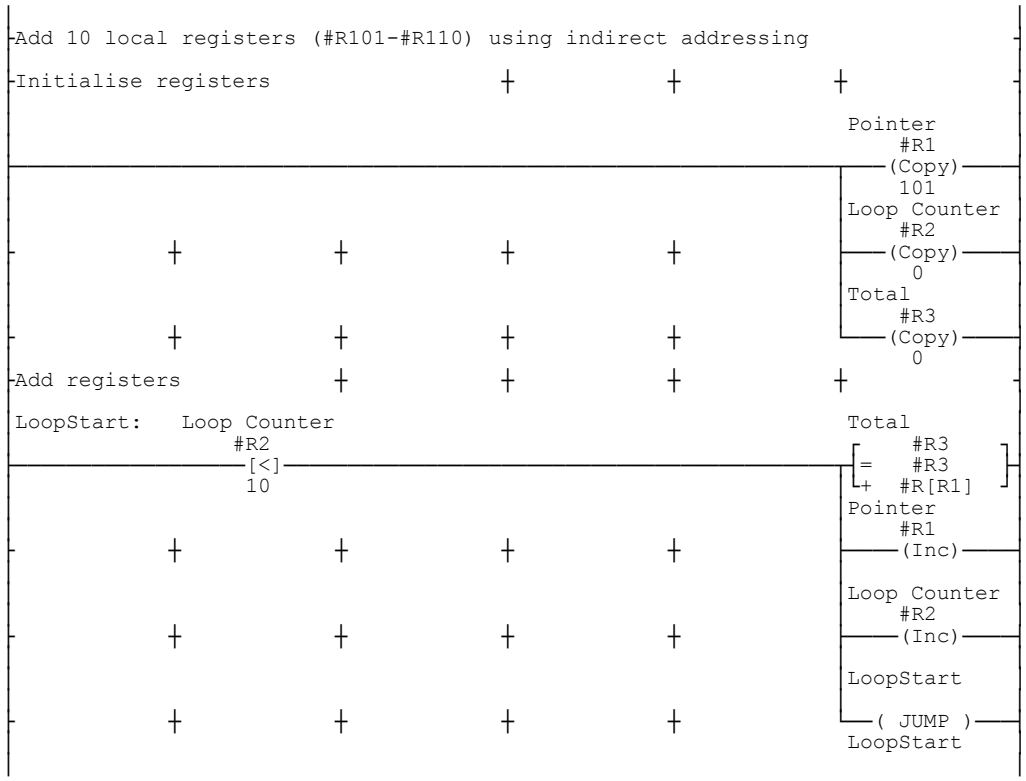


Figure: Adding 10 registers using indirect addressing

Example - Time Based Rolling Averages

This example counts the number of rainfall pulses each minute and then adds the total to a 60-register queue. Each minute a new value is added to the queue and the oldest value is removed and then the average value of all the one-minute totals is re-calculated. This provides the average rainfall per minute for the last hour. The amount of ladder logic required is greatly reduced by using indirect addressing as shown below.

The following registers are used:

#R498	Current number of rain pulses for this minute
#R499	Total number of rain pulses for the last minute
#R500	Average rainfall (pulses) per minute for the last 60 minutes (updated every minute)
#R501-560	Last 60 rainfall totals, oldest to newest totals respectively
#R1001	Queue pointer (pointer to next register to add)
#R1002	Loop counter
#R1003	Accumulated total (for averaging)

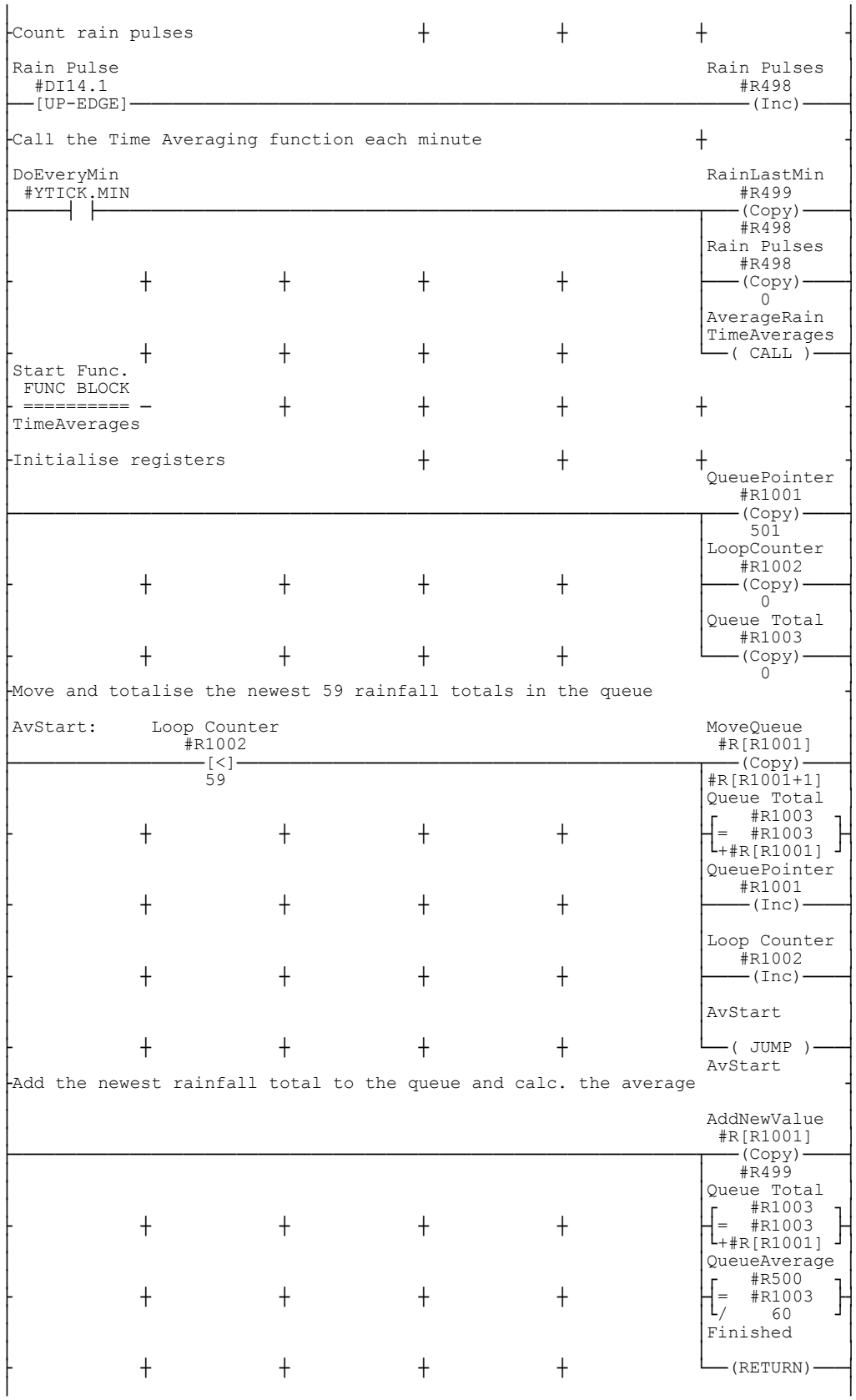


Figure: Averaging 60 rainfall totals using indirect addressing

Example - Polling RTUs Using A Function Block

The amount of ladder logic required to poll a number of RTUs can be greatly reduced by using a function block as shown below.

The function block allows the RTU address (%1) and the polling period in seconds (%2) to be specified. The RTU is only polled if a successful exception report has not been received for the last *period* seconds. The registers that are exception reported should be the same as the registers that are polled. The *PollRoutine* function block will poll the same registers from each RTU as defined in the RX Data block (these may need to be adjusted).

The following registers are used:
#NRrrr.63 The last value of the success counter for RTU *rrr*
#NRrrr.64 Quiet Timer. The amount of seconds since the last successful message to or from RTU *rrr*.
#R200 Temporary storage of the RTU address

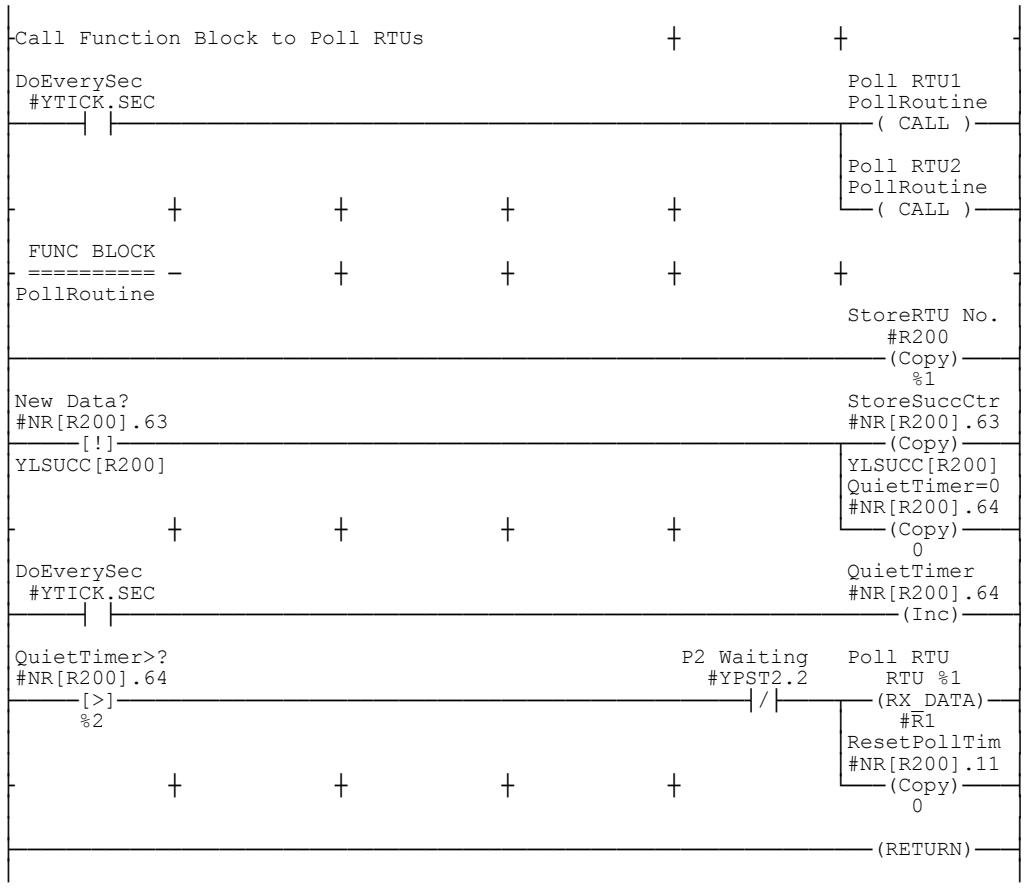
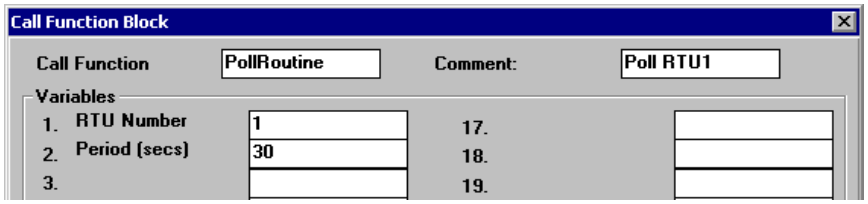
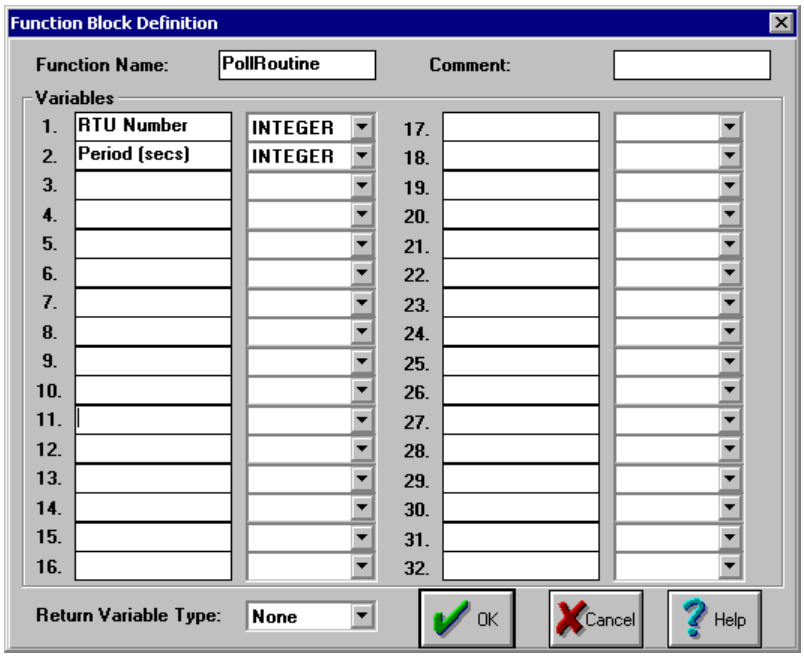
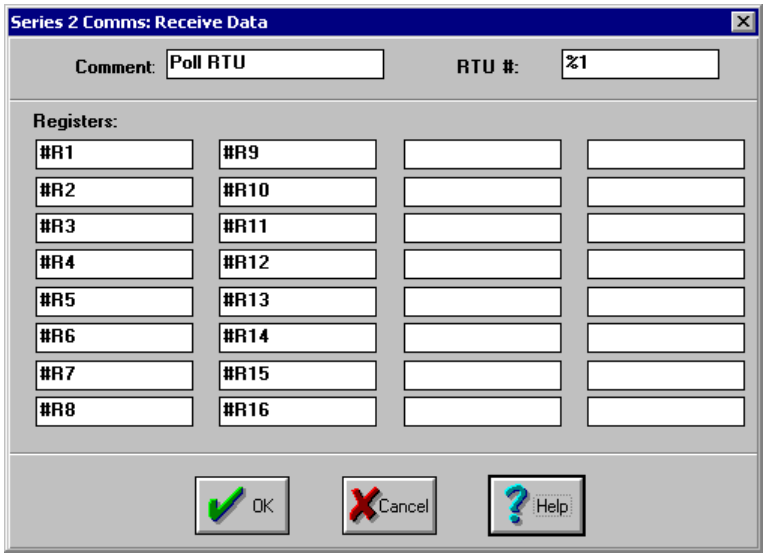


Figure: RTU polling function block

The details for the various ladder blocks are shown below.





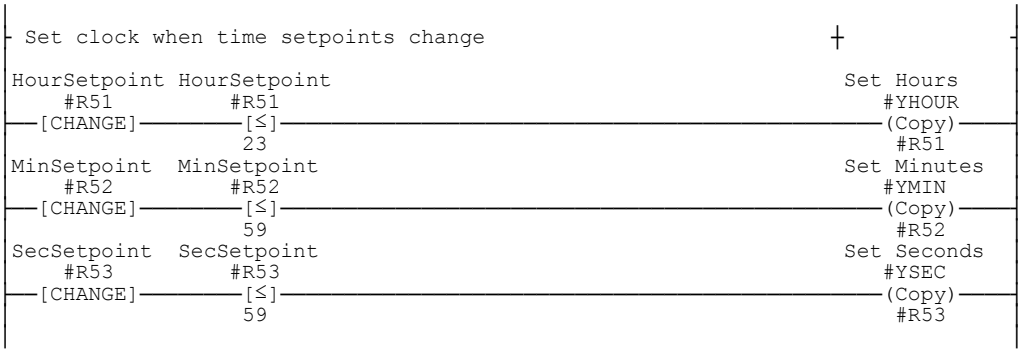


Example - Synchronizing RTU Clocks

Each RTU has a clock that can be synchronized periodically to ensure that data is logged with a precise time stamp.

There are various factors that affect the accuracy of clock synchronizing: drifting of the RTU's own clock, transmission time to set the clock, and accuracy of the source clock.

- The most accurate way to maintain clock accuracy is to use a CP-21 GPS option board. This allows the RTU's clock to be set within 10 milliseconds (10 thousandths of a second) of GMT. Please see the topic Driver - GPS.
- RTUs are also able to synchronize time between themselves. A master RTU can send a command to one or more outstations to synchronize all their clocks to its own clock. The RTU clocks will then be accurate to within 15 milliseconds. Please see the topic Ladder Logic - Clock Synchronization for details.
- An RTU clock can be manually updated by setting the hours, minutes, seconds and milliseconds registers of the RTU clock using Toolbox or SCADA software. Accuracy is dependent on the person or device used to set these registers. An example of doing this is shown below.



- RTUs also recognise a DNP3 or Kingfisher protocol function code that allows their clock to be set to millisecond accuracy.

Downloading Ladder Logic

Ladder logic must be compiled before it can be downloaded into the RTU. This is performed from the **Ladder/Logic, Compile** menu. After a successful compilation, the window below will be displayed. Note: if using RTU firmware prior to V1.30A, the firmware version will need to be changed for the ladder compiler from the menu **Ladder/Logic, Target Firmware Version.**

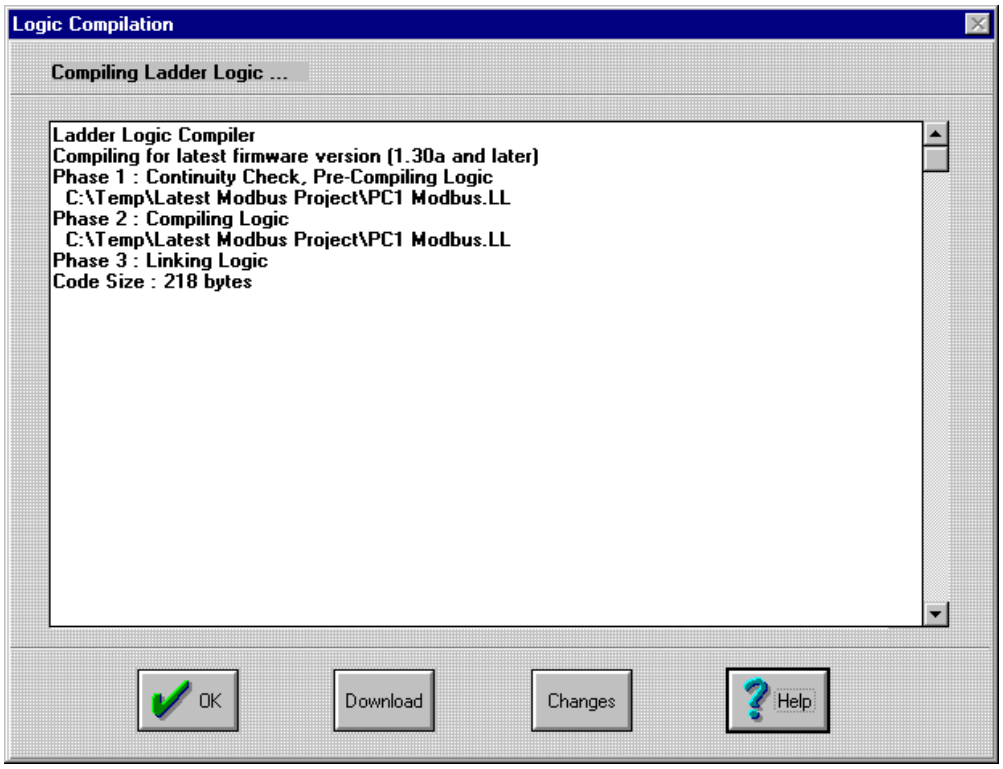


Figure: Window displayed when ladder logic is compiled.

If compilation is not successful, then the errors and the rung number of the error will be listed in the window.

After the ladder has been successfully compiled, select **Ladder, Download To RTU**. If downloading is successful, the message *Download Complete!* will be displayed otherwise *Comms Failure* will be displayed. Note: the SDB site configuration file must be downloaded first.

Download Changes to RTU

This option allows the user to only download changes since the last download. It is faster than performing a full download and is very useful when dealing with a large amount of logic or a slow communications link. Note: whenever an RTU is cold started or its memory configuration is changed, ladder logic is erased. It is then necessary to perform a *complete* download of ladder logic.

When ladder logic is compiled, an LLO file is created (ie. Filename.LLO). When the complete ladder logic configuration is successfully downloaded, Toolbox copies the LLO file and makes an LLX file (Filename.LLX). The LLX file is used to denote the ladder logic that is in the RTU. When downloading changes, Toolbox compares data blocks between the new LLO file and the last LLX file. Any new data blocks are then downloaded and then the LLO file is copied over the LLX file again.

Whenever ladder changes are downloaded to the RTU, a CRC is also downloaded. The RTU calculates its own CRC and compares it with the downloaded CRC. If they do not match, a *Ladder Error* flag is triggered and the ladder logic will be disabled. The success of the download changes can be checked by viewing the RTU Status. If an error has occurred, *Ladder Error* will be displayed and ladder logic will be disabled. To fix a *Ladder Error* flag, download the *complete* ladder.

Uploading And Downloading The Ladder Edit File

It is possible to store the ladder edit file (Filename.LL) in RTU memory so that future edits can be made by uploading the ladder edit file from the RTU**,** performing the edits and then downloading the compiled ladder logic and the new ladder edit file. When the ladder edit file is stored in the RTU, the date, time and file name are also stored. This allows the file version to be checked using **Logic, Advanced, Upload .LL File Details From RTU** (note: the **Logic** menu is available when the SDB site configuration window is in selected).

The Ladder Edit File is automatically stored in the RTU whenever ladder logic is downloaded if the Store ladder logic files in RTU check box is ticked in the memory configuration or it can be manually downloaded using **Logic, Advanced, Download .LL File To RTU**.

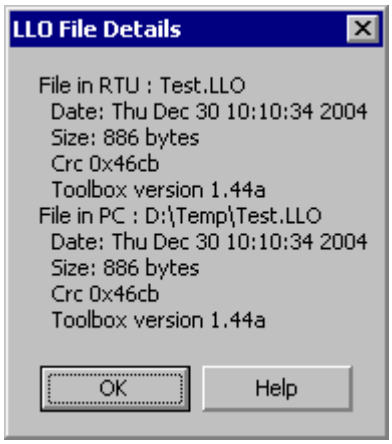
Before a ladder edit file can be uploaded, an RTU configuration (filename.SDB) must be open in Toolbox. Toolbox then uses the name of this configuration file for the uploaded ladder logic. The RTU configuration file can be uploaded from the RTU (using **Utilities, Advanced, Upload Configuration**) or alternatively an RTU configuration can be opened or created that does not have any ladder logic.

Once an RTU configuration file is open (if the configuration is uploaded from the RTU it must first be saved using the **File, Save As** command), the ladder logic can then be uploaded using **Logic, Advanced, Upload .LL File From RTU**.

Compare Ladder Version In RTU

When the SDB file is the active window (in front), the **Logic** menu option will appear. By selecting **Logic, Advanced, Compare ladder version in RTU**, the compiled ladder version in the RTU can be compared with the compiled ladder version in the PC.

To check if the ladder edit file (Filename.LL) in the PC corresponds to the compiled ladder logic in the RTU, compile the ladder logic on the PC (**Logic, Compile**) and then select **Compare ladder version in RTU**. This will compare the compiled ladder logic file (Filename.LL0) file in the PC with the compiled ladder logic file in the RTU. If the Size and CRC of the two files match, then all is OK as illustrated below. Different file dates are unimportant.



Date: Date and time when the ladder logic was compiled.

Size: Number of bytes in the compiled ladder file. Since comments are not included in the compiled file, changing comments and re-compiling will not affect the LLO file size.

Crc: Returns the hexadecimal value of the CRC (*0x* denotes hexadecimal). Since comments are not included in the compiled file, changing comments and re-compiling will not affect the CRC. Note: Always returns 0 when compiled and downloaded using Toolbox 1.41n and older.

Toolbox Version: Toolbox version used to generate the compiled ladder logic file.

Debugging Ladder Logic

To allow debugging of ladder logic, it is possible to view the state of the various ladder blocks and the contents of registers. This is accessed from the menu **Ladder, Debug**. Blocks that are logically true are shown in red. Debug simply reads all the data values from the RTU and displays all the current states and values in the ladder. Debug does not sequentially step through the ladder performing one rung, reading the data then performing the next rung and reading the data etc. Debug simply shows the current value of each address after the complete ladder has been processed. Note: registers can be viewed in unsigned or hexadecimal format. Please see View, Local Registers for details.

7. Communication Drivers

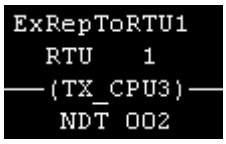
Drivers detailed in this chapter: Series 1, Omron PLC, Allen Bradley, Inline Flow Computer, Fuji PLC, ASCII, Hart, GPS, User Defined, Modbus, Trio E Series Radio and Spread Spectrum Radio.

For a comprehensive list of protocols and functions that are supported by Kingfisher RTUs, please see the document **protocols.pdf** available from http://helpdesk.servelec-semaphore.com.

Driver - Kingfisher Series I

Kingfisher Series I RTUs use two types of processor modules - CPU3 or CPU1. The following two ladder blocks are used for both processor types.

A Series I RTU has two possible locations for data, its SDT (scan data table) and its NDT (network data table). An SDT is comprised of 240 registers called *IDs* while the NDT is comprised of 240 blocks of 30 registers (7200 network registers in total). When a *Series 2* RTU receives data from a *Series I* RTU, it is always stored in network registers.



TX Series 1

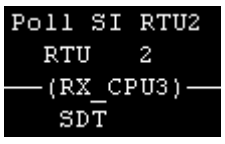
Transmits up to 30 Series 2 local registers to a Kingfisher Series I RTU.

Comment: A 12-character description.

RTU #: (1-255) The destination RTU that the data is sent to (note: addresses 250-255 are reserved for paging and PC use).

NDT No. (1-240; 0=SDT): When sending data to the NDT of a Series I RTU, usually *NDT No.* is chosen to correspond to the local RTU's own address. Eg. If RTU2 was sending data to Series I RTU1, NDT No. would be 2. When data is transferred to the SDT, registers are stored at the IDs corresponding to each register number (ie. R1 is stored at ID1, R2 is stored at ID2 etc).

Registers: Only local registers R1 to R240 can be transferred to a Series I RTU as series I RTUs only have 240 IDs (each ID is equivalent to a local register). To transfer hardware registers (#AI, #AO, #DI, #DO) or network registers (#N), these must first be copied into local registers (#R1 to #R240).



Rx Series 1

Polls up to 30 Series 1 IDs or an NDT block from a Kingfisher Series I RTU.

Comment: A 12-character description.

RTU #: (1-255) The source RTU that the data is read from (note: addresses 250-255 are reserved for paging and PC use).

NDT No. (1-240; 0=SDT): When *NDT No* is set to 0, data is read from the SDT otherwise it is read from the NDT block specified.

Registers: Only local registers R1 to R240 can be read from a Series I RTU as Series I RTUs only have 240 IDs (each ID is equivalent to a local register). When polling an NDT from a Series I RTU, it is not necessary to specify any registers as the 30 values of the NDT block are copied to 30 network registers of the NDT block number. The network registers used correspond to the IDs in the NDT block.

Sending Series I Data To A Series 2 RTU

IDs that are received from a Series I RTU are stored in network registers corresponding to the Series I RTU address. When an NDT block is received from a series I RTU, it is stored in the network registers corresponding to the NDT block number. The four types of Series I data transfers and the data storage locations are detailed below.

- SDT to SDT: IDs are stored in network registers corresponding to the IDs defined in the "Set Values Per ID Bank".
- SDT to NDT: IDs are stored in network registers corresponding to the IDs defined in the "Get Values Per ID Bank".
- NDT to NDT: IDs are stored in network registers corresponding to the IDs in the Series I NDT block.
- NDT to SDT: IDs are stored in network registers corresponding to the IDs defined in the "Set Values Per ID Bank".

Eg. Get ID Bank 1 contains IDs 1 and 31, Set ID Bank 2 contains IDs 201 and 202, NDT3 contains the IDs 61 and 62 and the Series I RTU sending the data is RTU2.

Series I Data Transfer	Get Bank (1,31)	Set Bank (201,202)	NDT Override	Series 2 Storage Locations
SDT to SDT	1	2	0	#NR2.201, #NR2.202
SDT to NDT	1	0	0	#NR2.1, #NR2.31
NDT to NDT	0	0	3	#NR3.61, #NR3.62
NDT to SDT	0	2	3	#NR3.201, #NR3.202

Polling Series I Data From A Series 2 RTU

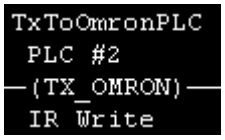
When IDs are polled from a Series 2 RTU, the Series 2 RTU will reply with its own local registers (R1 to R240) corresponding to the ID numbers. When an NDT is polled from a Series 2 RTU, the Series 2 RTU will reply with the first 30 values from the network registers corresponding to the NDT number.

Communicating With A Kingfisher Series I RTU

When using a TX_CPU3 or an RX_CPU3 block the RTU should be configured as follows:

- First ensure that the latest firmware and Kingfisher Series 1 driver are loaded in the RTU (as detailed in **protocols.pdf** available from http://helpdesk.servelec-semaphore.com).
- Set the port *protocol* as S1 Ctrl [controller] or S1 Outstn [outstation] in **Configuration, Port List** depending on whether the Series 2 RTU is to act like a master or outstation Series I RTU. Note: the "No CRC" port protocol option should only be selected when communicating with a CPU1 module over an RS232 line.
- Set the port *baudrate* in **Configuration, Port List**.
- Add the address of the Series I RTU to the network list in **Configuration, Network List** (this must be a unique address in the RTU network). Specify a system ID of AC Hex (for a CPU3 or CPU1 with version 3 EPROM) or A5 Hex (for CPU1 prior to version 3 EPROM).

Driver - Omron PLC



Tx Omron PLC

Transmits one or more consecutive local registers to one or more consecutive addresses in an Omron PLC and then returns a response code from the Omron PLC. Also known as *Host Link* protocol.

Comment: A 12-character description.

PLC Unit no: (1-99) The Omron address. The local RTU treats an Omron PLC as if it is another RTU in the network. This means that the Omron PLC's address must be a unique RTU address in the Network List.

Command: The Omron area to send the data to. The available options are: IR (Internal Relay) Area Write, HR (Holding Relay) Area Write, AR (Auxiliary Relay) Area Write, LR (Link Relay) Area Write, and DM (Data Memory) Area Write.

RTU Register: (#R1 to #R1024) The starting local register to transmit to the Omron PLC.

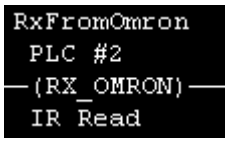
PLC Data Address: The starting address in the Omron PLC where the data is stored. The following addresses can be specified for each Command: IR Area Write: 0 to 235, HR Area Write: 0 to 99, AR Area Write: 0 to 27, LR Area Write: 0 to 063, DM Area Write: 0 to 999.

Data Length: (1-30) The Number of consecutive 16-bit registers to send.

Response Codes

The following response codes are stored in #NRxx.1 after each TX_OMRON block (where xx is the address assigned to the Omron PLC):

Response	Description
0	Normal completion
1	Not executable in RUN mode
2	Not executable in MONTOR mode
3	Not executable with PROM mounted
4	Address over (data overflow)
11	Not executable in PROGRAM mode
16	Parity error
17	Framing error
18	Overrun
19	FCS error
20	Format error (parameter length error)
21	Entry number data error (parameter error, data code error, data length error)
22	Instruction not found
24	Fame length error
25	Not executable (due to unclearable error, memory error, unwriteable EEPROM, missing IO table, etc)
34	No memory unit mounted
35	User memory is write protected
160	Aborted due to parity error in transmit data
161	Aborted due to framing error in transmit data
162	Aborted due to overrun in transmit data
164	Aborted due to format error in transmit data
165	Aborted due to entry number data error in transmit data.
168	Aborted due to frame length error in transmit data
Other	Probably produced by noise. Execute command again



Rx Omron PLC

Reads one or more consecutive Omron data words and stores them in the Kingfisher RTU's network registers.

Comment: A 12-character description.

PLC Unit no: (1-99) The Omron address. The local RTU treats an Omron PLC as if it is another RTU in the network and will store the Omron PLC data in network registers corresponding to this address. This means that the Omron PLC's address must be a unique RTU address in the *Network List*.

Command: The Omron area to read the data from. The available options are: Status Read, IR (Internal Relay) Area Read, HR (Holding Relay) Area Read, AR (Auxiliary Relay) Area Read, LR (Link Relay) Area Read and DM (Data Memory) Area Read.

PLC Data Address: The starting address in the Omron PLC where data is to be read from. The following addresses can be specified for each Command:

- Status Read: 1
- IR Area Read: 0 to 235
- HR Area Read: 0 to 99
- AR Area Read: 0 to 27
- LR Area Read: 0 to 63
- DM Area Read: 0 to 999 (read/write) and 1000 to 1999 (read only)

RTU Register: (#R2 to #R1024) Specified as a local register but indicates the first network register to begin storing the data from. Note: the first network register is reserved for the response code returned by the PLC after each message.

Data Length: (1-30) The Number of consecutive 16-bit registers to read.

Communicating With An OMRON PLC

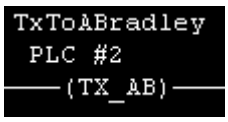
When using a TX_OMRON or an RX_OMRON block the RTU should be configured as follows:

- First ensure that the Omron driver is supported by the type of RTU that is being used and that the latest firmware and driver are loaded in the RTU (as detailed in **protocols.pdf** available from http://helpdesk.servelec-semaphore.com).
- Set the port *protocol* as OMRON in **Configuration, Port List**.
- Set the port *baudrate* in **Configuration, Port List**.
- Add the station address of the PLC to the network list in **Configuration, Network List** (this must be a unique address in the RTU network).

Driver - Allen Bradley

Kingfisher RTUs communicate with Allen Bradley PLC's using the DF1 protocol in half duplex slave mode with a 2-byte CRC. The data format is 8 data bits, 1 stop bit and no parity.

The Allen Bradley PLC returns an error code after each message. The error code is stored in the first network register corresponding to the address assigned to the PLC. The STS error code is stored in the lower 8 bits of the register and the EXT STS error code in the high 8 bits. A valid message will reset the value to 0.



Tx Allen Bradley

Transmits up to 100 consecutive local registers from a Kingfisher RTU to an Allen Bradley PLC (PLC5 or SLC500).

Comment: A 12-character description.

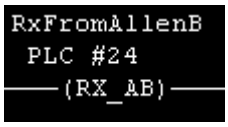
Station No. (decimal): (1-249) Station address configured in the Allen Bradley PLC. An Allen Bradley PLC is treated like another RTU in the network. This means that the station address must be different to all the other RTU addresses in the RTU's Network List.

Destination Register: Destination address in the Allen Bradley PLC where data is stored. This should be a string reference like N10:1.

Source Register: (#R1 to #R1024) Local register of the RTU to read the data from.

No. Of Registers: (1-100) Number of 16-bit registers to send. Note: each floating point value uses two 16-bit local registers. The maximum number of bytes that a message sent by an RTU can contain is 250. The number of bytes sent varies depending on the contents of the message. In rare cases, the TX-AB message can exceed this limit and so the message is not sent and an error code of 65535 is returned in the first network register of the corresponding *Station No* RTU. It is therefore recommended that a maximum of 50 registers be written to the PLC in one message.

PLC Type: PLC5 / SLC500.



RX Allen Bradley

Receives up to 100 consecutive registers from an Allen Bradley PLC (PLC5 or SLC500).

Comment: A 12-character description.

Station No. (decimal): (1-249) Station address configured in the Allen Bradley PLC. An Allen Bradley PLC is treated like another RTU in the network. This means that the station address must be different to all the other RTU addresses in the RTU's Network List.

Destination Register: (2 - 1024) Network register of the Kingfisher RTU to begin storing the data from. Note: the first network register is reserved for the error code returned by the PLC after each message.

Source Register: Source address in the Allen Bradley PLC where data is read from. This should be a string reference like N10:1.

No. Of Registers: (1-100) Number of 16-bit registers to read. Note: each floating point value uses two 16-bit local registers.

PLC Type: PLC5 / SLC500.

Communicating With An Allen Bradley PLC

When using a TX_AB or an RX_AB block the RTU should be configured as follows:

- First ensure that the Allen Bradley driver is supported by the type of RTU that is being used and that the latest firmware and driver are loaded in the RTU (as detailed in **protocols.pdf** available from http://helpdesk.servelec-semaphore.com).
- Set the port *protocol* as ALLEN BRADLEY in **Configuration, Port List**.
- Set the port *baudrate* in **Configuration, Port List**.
- Add the station address of the PLC to the network list in **Configuration, Network List** (this must be a unique address in the RTU network).

Note 1: The simplest way to connect between an RTU serial port and an Allen Bradley PLC is to use an RS232 null modem cable (can use the Semaphore ADP-05 adapter and an RJ45 to RJ45 lead). The Allen Bradley PLC requires a RS232 DB9 male port. An Allen Bradley SLC5/**03** CPU has a DB9 male port while the SLC5/**02** CPU only has an RS485 RJ45 port and must have a communications module installed.
Note 2: if using Ethernet to communicate, the Allen Bradley driver uses TCP/IP port 2222 to communicate with the Allen Bradley PLC.
Note 3: If a 1785-KE interface module is used between the PLC and the RTU, the 1785-KE station number must also be configured in the network list. The PLC should then be configured as an indirect link via the 1785-KE station address.

Driver - Inline Flow Computer

Rx Inline

Receives one data parameter from an Inline flow computer.

Comment: A 12-character description.

RTU #: (1-255) The RTU address assigned to the Inline flow computer (note: addresses 250-255 are reserved for paging and PC use).

Command: The data parameter to read. The available options are: read RTC (real-time clock), read pressure, read temperature, read corrected volume, read uncorrected volume, read correction factor, read maximum hourly demand, read maximum daily demand, read current hourly usage, read current daily usage, read volumes at end of the day and read flow rate.

The data returned from the Inline flow computer is placed in the network registers of the corresponding RTU address. The network registers that are used for each command are shown below.

Command	Received Data	
Read real-time clock	#R1	seconds, 0-59
	#R2	minutes, 0-59
	#R3	hours, 0-23
	#R4	day of week, 1-7 (1 = Sunday)
	#R5	day of month, 1-31
	#R6	month, 1-12
	#R7	year, 0-99
Read correction factor	#R9	least significant word of correction factor
	#R10	most significant word of correction factor
Read pressure	#R11	least significant word of pressure
	#R12	most significant word of pressure
	#R13	pressure flags
	#R13.8	1=Manual, 0=Auto
	#R13.7	1=Error
	#R13.6	1=Gauge, 0=Absolute
	#R13.5	1=PSI, 0=kPa
	#R13.4	decimal point: 1 = x 0.01 , 0 = x 0.1
Read temperature	#R15	most significant word of temperature
	#R16	least significant word of temperature
	#R17	temperature flags
	#R17.8	1=Manual, 0=Auto
	#R17.7	1=Error
	#R17.6	1=Deg F, 0=Deg C
Read corrected volume	#R19	least significant word of corrected volume
	#R20	most significant word of corrected volume
	#R21	corrected volume flags
	#R21.8	1=cubic feet, 0=m3

Command	Received Data	
Read uncorrected volume	#R23	least significant word of uncorrected volume
	#R24	most significant word of uncorrected volume
	#R25	uncorrected volume flags
	#R25.8	1=cubic feet, 0=m3
Read corrected volume flow rate	#R27	least significant word of flow rate.
	#R28	most significant word of flow rate.
	#R29	daily usage flags
	#R29.8	1=cubic feet, 0=m3
Read volumes at end of the day	#R31	day of month of volumes
	#R32	month of the volumes
	#R33	least significant word of corrected volume
	#R34	most significant word of corrected volume
	#R35	least significant word of uncorrected volume
	#R36	most significant word of uncorrected volume
	#R37	volume flags
	#R37.8	1=cubic feet, 0=m3
Read maximum hourly demand	#R40	hour of maximum hourly demand
	#R41	day of week of maximum hourly demand
	#R42	day of month of maximum hourly demand
	#R43	least significant word of maximum hourly demand
	#R44	most significant word of maximum hourly demand
Read maximum daily demand	#R45	day of week of maximum daily demand
	#R46	day of month of maximum daily demand
	#R47	least significant word of maximum daily demand
	#R48	most significant word of maximum daily demand
Read current hourly usage	#R51	least significant word of hourly usage
	#R52	most significant word of hourly usage.
	#R53	hourly usage flags
	#R53.8	1=cubic feet, 0=m3
Read current daily usage	#R55	least significant word of daily usage
	#R56	most significant word of daily usage.
	#R57	daily usage flags
	#R56.8	1=cubic feet, 0=m3

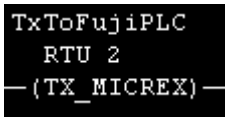
Communicating With An Inline flow computer

When using an RX_INLINE block the RTU should be configured as follows:

- First ensure that the Inline driver is supported by the type of RTU that is being used and that the latest firmware and driver are loaded in the RTU (as detailed in **protocols.pdf** available from http://helpdesk.servelec-semaphore.com).
- Set the port *protocol* as INLINE or INLINE2 in **Configuration, Port List**. INLINE2 is the newer Inline protocol. Note: the Inline driver must be configured on a serial port.
- Set the port *baudrate* in **Configuration, Port List**.
- Add the Inline flow computer address to the network list in **Configuration, Network List** (this should be a unique address in the RTU network). Also set the message time out to at least 3000 ms.
- Ensure that the RTU has the correct time and date to allow the RTU to read the volumes at the end of the day.

Driver - Fuji PLC

There are currently 2 Fuji PLC drivers - Micrex-F and NJ Series. The following information applies to both PLC types.



Tx Micrex / Tx Fuji NJ

Transmits up to 58 consecutive local registers from a Kingfisher RTU to a Fuji PLC. Returns a response code in the first network register corresponding to the address assigned to the Fuji PLC.

Comment: A 12-character description.

PLC number: (1-249) The RTU address assigned to the Fuji PLC.

RTU Source: (#R1 to #R2048) The starting local register to transmit to the Fuji PLC.

PLC Destination: The data address in the Fuji PLC to write the data to. Eg. K0

Number of registers: (1-58) Number of registers to be written to the Fuji PLC.

Rx Micrex / Rx Fuji NJ

Reads up to 106 consecutive registers from a Fuji PLC and stores them in the RTU's network registers. Returns a response code in the first network register corresponding to the address assigned to the Fuji PLC.

Comment: A 12-character description.

PLC number: (1-249) The RTU address assigned to the Fuji PLC.

PLC Source Address: The starting address in the Fuji PLC from where the data is read. Eg. BD12.

RTU Destination: (#R2 to #R1024) Specified as a local register but indicates the first network register to begin storing the data from. Note: the first network register is reserved for the response code returned by the PLC after each message.

Number of registers: (1-58) Number of registers to read.

Response Codes (hexadecimal values)

Fuji Micrex-F

00 Processing is completed normally.
12 Data is written in program area.
20 Specified command code does not exist.
21 Input data does not sequence correlation to command. Example: Read or write by not using 4 byte units for 32-bit area.
22 Operation is available by loader only.
24 Module designated does not exist.
32 Oversized address for model number is designated.
A0 Communication module number error.
A1 Communication module busy.
A2 Exceeded number of data.
A3 Not suitable communication module.
A6 No data module.
A7 Unidentified boundary of data module to be transferred with receive byte number.
A8 Command error

Fuji NJ PLC

00 Processing is completed normally.
01 Data is written to ROM.
02 Specified command code does not exist.
03 Inconsistent data (parameter error).
04 Processing is impossible due to transmission interlock by another device or loader.
05 Incorrect module No.
06 Search item not found.
07 An address exceeding the module's range was specified (during writing)
08 An instruction error was found in a writing program.
09 Program execution cannot continue due to an error.
0A After sending a WAK, a command other than cancel or continue was received.
0B Mismatched loader type (a command was sent to a loader that cannot be connected to the NJ series PLC).
0C Mismatch password.
0E Connection to network is impossible.
0F Another loader is communicating over the network.
21 Now processing.
A7 Transmission error.

Communicating With A Fuji PLC

When using a TX/RX_MICREX or TX/RX_FUJI_NJ block the RTU should be configured as follows:

- First ensure that the Fuji driver is supported by the type of RTU that is being used and that the latest firmware and driver are loaded in the RTU (as detailed in **protocols.pdf** available from http://helpdesk.servelec-semaphore.com).
- Set the port *protocol* as Fuji Micrex-F or Fuji NJ-Series in **Configuration, Port List**.
- Set the port *baudrate* in **Configuration, Port List**. A serial port is used to communicate with a Fuji PLC.
- Add the address of the PLC to the network list in **Configuration, Network List** (this must be a unique address in the RTU network).
- The Kingfisher RTU uses 8 data bits, no parity, 1 stop bit and CRC checking for communications with the Fuji controller. To enable CRC checking on the communication messages, the Fuji controller needs to be configured with an initialization file. The initialization table has to be defined in the ladder logic program and the message module registration has to be setup.

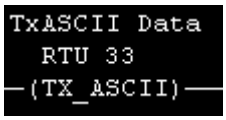
Driver - ASCII

The ASCII protocol allows a Kingfisher RTU to request information from an external device using an ASCII or hexadecimal protocol. Data returned from the device is stored in network registers corresponding to the address assigned to the ASCII device (note: an LP-1/2/3 stores the data in local registers).

The ASCII driver transmits a zero terminated string and then stores the received string in registers or scans the string for floating point variables.

Each character is stored as an 8-bit ASCII number. Two characters are stored in each local register. For each pair of characters, the left character is stored in channels 1-8 and the right character is stored in channels 9-16. For an LP-1/2/3, the characters are stored in reverse order.

The string sent to the external device can be a fixed string entered in the ladder logic or a string stored in local registers. The string must be zero terminated and can therefore not contain any zeros. The difference between the RX_ASCII and TX_ASCII ladder functions is that the TX_ASCII ladder function sends a byte string and returns a byte string. The RX_ASCII ladder function sends a byte string and returns a number of floating point variables. The received message string is scanned for any digits. Each set of digits is converted to a floating point value and stored in the network registers corresponding to the device address. The number of floating point values returned is configured in the ladder block. If less decimal numbers are found in the message than floating point values configured, the remaining floating point values are returned as zero.



Tx ASCII

Comment: A 12-character description.

RTU Number: (1-249) The RTU address assigned to the ASCII device.

String: Characters to be sent or first local register (#R) where string is stored.

Max. bytes in reply: (1-200) Maximum number of bytes to return.

Reply # bytes: (#R1-#R2048) Specified using a local register but indicates the network register where the number of bytes received is reported. Note: an LP-1/2/3 stores the data in local registers instead of network registers.

Reply destination: (#R1-#R2048) Specified using a local register but indicates the first network register where the received message bytes are reported. Note: an LP-1/2/3 stores the data in local registers instead of network registers.

Rx ASCII

Comment: A 12-character description.

RTU Number: (1-249) The RTU address assigned to the external ASCII device.

String: Characters to be sent or first local register (#R) where string is stored.

Destination: (#F1 to #F2047) First network float register where received floating point variables are reported. Specified using a local float register. Note: an LP-1/2/3 stores the data in local registers instead of network registers.

No. of variables: (1-40) Number of floating point variables contained in the reply message. Can also specify a local register (#R) that contains the *No. of variables* setting.

Communicating With An ASCII Device

When using a Tx or Rx ASCII block the RTU should be configured as follows:

- First ensure that the ASCII driver is supported by the type of RTU that is being used and that the latest firmware and driver are loaded in the RTU (as detailed in **protocols.pdf** available from http://helpdesk.servelec-semaphore.com).
- Set the port *protocol* as ASCII No Parity or ASCII Even Parity in **Configuration, Port List**. The ASCII protocol can only be used on MC or LP-1/2/3 ports.
- Set the port *baudrate* in **Configuration, Port List**.
- Add the address of the ASCII device to the network list in **Configuration, Network List** (this must be a unique address in the RTU network).

Driver Limitations

- Transmit string can not contain any zero bytes.
- Data format must be 8 bits, no or even parity and 1 stop bit.
- There is no validity checking of the reply message by the driver.
- Maximum length of a transmit string is 200 bytes.

Driver - Hart

The Hart protocol allows a Kingfisher RTU to request information from a Hart Field Communication device. Data returned from the Hart device is stored in network registers. The driver is based on Hart protocol revision 5.

Rx Hart

Comment: A 12-character description.

Hart Device Number: (0-15) The address of the Hart field device. Address 0 is only used for point to point installations.

Command: The following commands are supported:
(Note: each command is followed by the Hart function code (FC). Some manufacturers have different descriptions for the various Hart commands. PV stands for *primary variable*).

Universal Commands	FC
Read Unique Identifier	0
Read PV [primary variable]	1
Read Current and % of range	2
Read Current and 4 dyn vars	3
Write Polling Address	6
Read Unique Identifier with Tag	11
Read Message	12
Read Tag, Descriptor, Date	13
Read PV Sensor Information	14
Read [PV] Output Information	15
Read Final Assembly Number	16
Write Message	17
Write Tag, Descriptor, Date	18
Write Final Assembly Number	19

Common Practice Commands	FC
Read Transmitter Variables	33
Write [PV] Damping Value	34
Write [PV] Range Values	35
Set [PV] Upper Range Value	36
Set [PV] Lower Range Value	37
Reset Configuration Changed Flag	38
EEPROM Control	39
Enter/Exit Fixed Current Mode	40
Perform Transmitter Self Test	41
Perform Master Reset	42
Set PV Zero	43
Write PV Units	44
Trim [PV Current] DAC Zeros	45
Trim [PV Current] DAC Gain	46
Write [PV] Transfer Function	47
Read Additional Transmitter Status	48
Write PC Sensor Serial Number	49
Read Dynamic Variable Assignments	50
Write Dynamic Variable Assignments	51
Set Transmitter Variable Zero	52
Write Transmitter Variable Units	53
Read Transmitter Variable Info	54
Write Transmitter Variable Damping Value	55
Write Transmitter Variable Sensor Serial No	56
Read All Dynamic Variables	108

RTU: (1-255) The RTU address assigned to the Hart device.

Destination Register: (#R1 to #R2048) The network register where the return variables are stored.

Device Address (3 regs): (#R1 to #R2046) The first of 3 consecutive network registers that are used to store the extended address of the Hart device.

Source Register: (#R1 to #R2048) The network register where variables for writing operations are stored.

Status: (#R1-#R2048) Optional. The network register where the status and response codes of the Hart device are stored. Channels 1-8 = status code, channels 9-16 = response code.

Communicating With A Hart Device

When using an Rx Hart block the RTU should be configured as follows:

- First ensure that the Hart driver is supported by the type of RTU that is being used and that the latest firmware and driver are loaded in the RTU (as detailed in **protocols.pdf** available from http://helpdesk.servelec-semaphore.com). The RTU must also have a Hart option board installed on port 2 or port 3.
- Configure the port as follows (from **Configuration, Port List**): *Type*=RS-232, *Baud Rate*=1200, *Pre TX*=10ms, *Post TX*=15ms and *Protocol*=Hart. Note the Pre TX and Post TX times may need to be increased (eg. Post Tx=45ms) to suit the particular Hart device.
- Add the configured *RTU* address to the network list in **Configuration, Network List** (this must be a unique address in the RTU network).

Notes

- Data returned by commands 1, 2 and 3 is formatted and stored in Network registers as follows (where x = the local register number configured in *Destination Register*):

Command	Data Returned (#F = 32-bit float, #R = 16-bit register)
1	#Fx = Value, #Rx+2 = Units
2	#Fx = Primary Variable (PV) current, #Fx+2 = PV % of range
3	#Fx = Primary Variable current #Fx+2 = Primary Variable (PV) value, #Fx+4 = PV units #Fx+6 = Second Variable (SV) value, #Fx+8 = SV units #Fx+10 = Third Variable (TV) value, #Fx+12 = TV units #Fx+14 = Fourth Variable (FV) value, #Fx+16 = FV units

  Eg. Destination Register = #R101. For command 1, #F101 = value (uses #R101 and #R102) and #R103 = units.
- All other data written to and from the RTU will be in a raw format of 2 bytes per register, as returned by the external Hart Device.
- ASCII data, as returned by the Hart Protocol, is in a six-bit packed format and is stored in the RTU in this format, with 2 bytes per local register.

Driver - GPS

Rx GPS

For use with a CP-21 GPS option board (superseded). Allows a Kingfisher RTU to determine its location and synchronize its clock within 10 milliseconds of universal time anywhere in the world (GPS option board required).

Comment: A 12-character description.

Device #: (1-249) Address assigned to the GPS option board. Data returned from the GPS option board is stored in network registers corresponding to this address.

Update Freq (secs): (1-255, typically 1 sec) Setting determines how often the GPS option board sends the new GPS data to the RTU.

Time Offset (mins): (0-1440) Enter the time offset to the Greenwich mean time of the local RTU. (eg. Melbourne = GMT + 600 mins).

Synchronize Clock?: If ticked, the RTU real time clock is updated at the rate specified by the Update Frequency above.

Dest. For Position: (#R1 to #R2043) Specified as a local register but indicates the first network register to begin storing the position data from.
#Rx = degrees of latitude
#Fx+1 = float 0-59.999999 (6 decimal places) minutes of latitude
#Rx+3 = degrees of longitude
#Fx+4 = float 0-59.999999 (6 decimal places) minutes of longitude

Dest. For Time: (#R1 to #R2045) Specified as a local register but indicates the first network register to begin storing the time data from.
#Rx = hours (0-23)
#Rx+1 = minutes (0-59)
#Fx+2 = float 0-59.999 seconds

Dest. For Altitude: (#R1, #R3 … #R2047 odd registers) Specified as a local register but indicates the network float register to store the altitude data in. #Fx = float of altitude.

Dest. For #Satellites: (#R1 to #R2048) Specified as a local register but indicates the network register to store the number of satellites data in. This is the number of satellites that are currently in contact with the GPS option board.

Communicating With A GPS Option Board

When using a GPS option board the RTU should be configured as follows:

- First ensure that the GPS option board is supported by the type of RTU that is being used and that the latest firmware is loaded in the RTU (as detailed in **protocols.pdf** available from http://helpdesk.servelec-semaphore.com).
- Configure the port as follows (from **Configuration, Port List**): *Type*=GPS, *Baud Rate*=19200, *Pre TX*=0ms, *Post TX*=0ms and *Protocol*=GPS.
- Add the configured *Device* address and port number of the GPS option board to the network list in **Configuration, Network List** (this must be a unique address in the RTU network). Specify a *Timeout* of 2000ms.
- The GPS option board must be configured on the first ladder scan using the RX GPS block.

Driver - User Defined

The User Defined protocol allows new comms protocols to be developed completely in ladder logic. The User Defined protocol is similar to the ASCII protocol driver but is more useful because it is not limited to null-terminated ASCII strings, can accept unsolicited incoming messages and will work on all communication ports.

Note: Comms Analyser will only show User Defined protocol traffic after it has been processed by the function block. That is Logic must be enabled for Comms Analyser show any User Defined traffic.

Tx User

Transmits a string up to 200 characters long to an external device and then receives and stores the response string (if any). If any bytes are received, the success counter corresponding to the device number is incremented (Note: this protocol does not verify any checksum or CRC bytes in the message). If no bytes are received within the timeout period, the fail counter is incremented. Tx User also works with other port protocols (other than User Defined eg. Series 2). All characters received while the Tx User function is active, are treated as the Tx User response string.

Comment: A 12-character description.

Device Number: (1-249) The RTU address assigned to the external device. Note: the Device Number is only used to access the communications parameters stored in the Network List (and Phone List for PSTN devices); it does not have to correspond to the physical address of the external device. The network link configured for this Device Number is used to control communications.

TxData Source: (#R1 to #R2048) Local register containing the first character of the string to be transmitted.

Tx no. bytes: (#R1 to #R2048 or 1 to 200) The number of bytes of the string to transmit. Can be specified as a local register or a constant.

RxData destination: (#R1 to #R2048) First local register to begin storing the received string in.

Rx No. bytes (max): (#R1 to #R2048 or 0 to 250) Maximum number of bytes expected in the response. This can be specified as a local register or a constant. Enter 0 if no reply is expected. If non-zero, after the Tx message string has been sent, the RTU will wait for a reply. The RTU will stop waiting after the timeout specified in the Network List has expired. If a local register is specified, the register will be updated after the function has completed to correspond to the actual number of bytes received. Note: a maximum of 250 bytes can be received on a processor port.

Status Register: (#R1 to #R2048 or Blank) If a register is specified, it will be updated with the status of the Tx User function as follows:

- Ch1: Waiting flag. Set ON when the block is activated and set OFF when the block is finished.
- Ch2: Status flag. Written to after the block is finished. Set OFF if the update was successful or set ON if the update failed (due to communications failure).

Use network registers ?: If this box is selected, the function will read and write to network registers for the specified device instead of local registers. This applies to the Tx and Rx data strings, the number of bytes and the status register.

Local link ?: If this box is selected, the function will send the messages to the local device. When using PSTN, GSM, GPRS or similar communications, the Tx User message will be sent to the local modem itself. Eg. Local Link is used when reading SMS messages from a GSM or sending other AT commands.

The example shown below transmits a string of 10 characters from local registers starting at #R200 to the external device assigned to RTU100. A maximum of 50 characters are expected as a reply and the reply is stored in local registers starting at #R101. If any characters are received, success counter #YLSUCC100 is incremented, otherwise fail counter #YLFAIL100 is incremented. Note: if more than 50 characters are in the receive buffer, only the first 50 characters will be read.

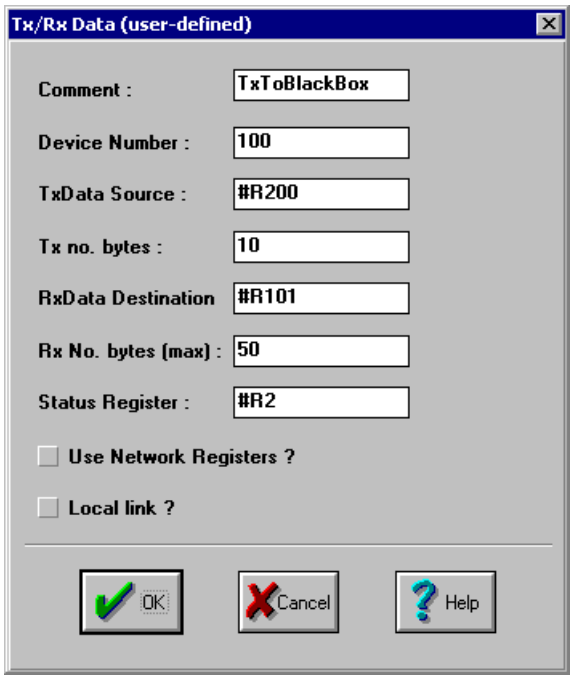


Figure: Example Tx User block

Rx User

Rx User allows an incoming message string to be received and stored. When bytes are received, the success counter corresponding to the device number is incremented (Note: this function cannot verify any checksum or CRC bytes in the message). If no bytes are received after triggering an Rx User block, the fail counter is incremented. Rx User will not work with other port protocols (always returns 0) because any characters received are processed as the other protocol.

The port register #YPRXCnn can be used to determine if there are any received characters/bytes in the buffer of port *nn* (nn=1 to 16). When #YPRXCnn is non zero, the *RxUser* function can then be used to retrieve the message and store it in local registers.

Comment: A 12-character description.

Device Number: (1-249) The RTU address assigned to the external device. Note: the Device Number is only used to access the communications parameters stored in the Network List (and Phone List for PSTN devices); it does not have to correspond to the physical address of the external device. The network link configured for this Device Number is used to control communications.

RxData destination: (#R1 to #R2048) First local register to begin storing the received string in.

Rx No. bytes (max): (#R1 to #R2048 or 0 to 250) Maximum number of bytes expected in the response. This can be specified as a local register or a constant. The RTU will attempt to receive the specified number of bytes. It will wait until the timeout specified in the Network List has expired. If a local register is specified, the register will be updated after the function has completed to correspond to the actual number of bytes received.

Status Register: (#R1 to #R2048 or Blank) If a register is specified, it will be updated with the status of the Rx User function as follows:

- Ch1: Waiting flag. Set ON when the block is activated and set OFF when the block is finished.
- Ch2: Status flag. Written to after the block is finished. Set OFF if the update was successful or set ON if the update failed (due to communications failure).

Use network registers ?: If this box is selected, the function will read and write to network registers for the specified device instead of local registers. This applies to the Rx data string, the number of bytes and the status register.

The example shown below continuously monitors port 2 for messages. When any characters are received and the previous Rx User command has finished (tests the configured Status Register #R2.1), the *Rx User* function reads up to 100 characters from the port 2 comms buffer. These characters are then stored in local registers starting at #R120 and then the actual number of received characters are reported in #R1. Success counter #YLSUCC100 is incremented each time characters are received. RTU100 is configured as *Direct via Port 2* in the Network List.

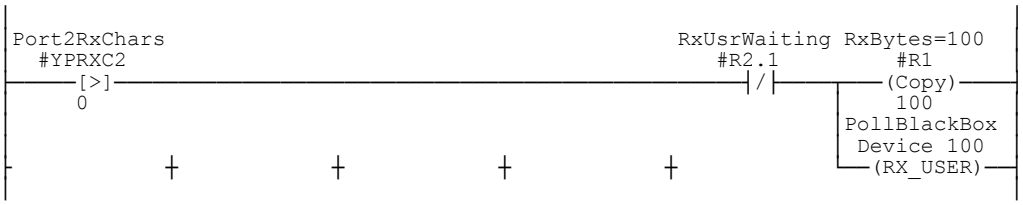


Figure: Example ladder logic used to receive characters

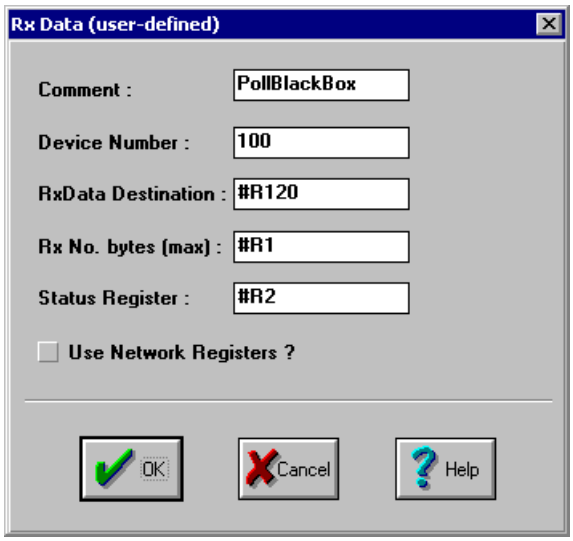


Figure: Rx User block used in Figure 6.9b.

Handling Messages of Unknown Size:

There are two different ways of receiving messages of unknown size:

- Specify the maximum possible message size in the *Rx No. bytes* field, and allow the RTU to report the actual number of characters received.
- Initially specify just the first few bytes of the message in the *Rx no. bytes* field. Most comms protocols include a header at the start which can be decoded to work out the complete message size. When this has been done (in ladder logic), the *Rx User* function can be called again, specifying the actual expected size of the remaining part of the message, and if necessary specifying a new *RxData Destination*.

Communicating With A User Device

When using a Tx or Rx User block the RTU should be configured as follows:

- First ensure that the User Defined protocol is supported by the type of RTU that is being used and that the latest firmware and driver is loaded in the RTU (as detailed in **protocols.pdf** available from http://helpdesk.servelec-semaphore.com).
- Configure the port for the correct *baudrate* and set the port *protocol* to User-Defined (not necessary for Tx User). After selecting User Defined, another configuration box will appear. The number of data bits (7 or 8), the parity (none, even or odd) and the number of stop bits (1 or 2) can then be specified (default setting: 8 data bits, no parity, 1 stop bit). Note: the Tx User function can be used with any other port protocol eg. Series 2 or paging. When using multiple protocols on the same port, Tx User messages must only be initiated when the port is completely free (pager messages must also be completed - ie. ensure #YLST250.2 is OFF) otherwise communication errors will occur.
- Assign an RTU address to the external device and add this to the network list. Configure the port number, the timeout and the number of retries to be used when communicating with the external device.

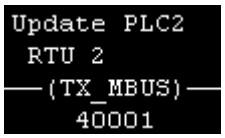
Error Checking

No validity checking (eg. checksum or CRC) of incoming messages is performed by the *RxUser* or *TxUser* functions. Messages are accepted regardless of any CRC or checksum bytes and so message integrity must be checked using ladder logic.

Driver - Modbus

The Modbus driver allows the RTU to respond to and initiate Modbus messages. If the RTU only needs to respond to messages (eg. from an operator panel or PC) then the Tx Modbus and Rx Modbus ladder blocks are not used.

Kingfisher RTUs use a data format of 8 data bits, no parity bit and 1 stop bit and support the *RTU* Modbus data format. Note: an MC-xx module can also initiate Modbus messages in ASCII format (7 data bits, even parity and 1 stop bit). Modbus is also supported on processor Ethernet ports (CP-10/11, CP-21).



Tx Modbus

Transmits 16-bit registers or digital channels to a Modbus device.

Comment: A 12-character description.

RTU #: (#R1 to #R2048 or 1 to 249) The destination RTU (or PLC) address. Can be specified as a local register or a constant.

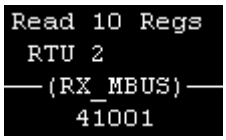
Register: The starting register or bit (channel) to send to the destination RTU. Allowable *Register* settings are: #Rxxxx, #Rxxxx.cc where xxxx=local register number (1-2048) and cc=channel number (1-16). Network registers (#N…) cannot be used. Note: before sending floating point or long registers to a Modbus device, the 2 Kingfisher local registers used to store the number will need to be reversed. Please see *Modbus Floating Point and Long Registers* below for more information.

MODBUS Address: (1 to 49999 or L000001 to L465535 when using extended addressing) The starting address to store the registers from in the destination PLC (or RTU). The registers are stored at consecutive addresses starting from the address specified here.

No. of Points: (#R1 to #R2048 or 1 to 123) The number of consecutive 16-bit registers or channels (single bits) to send. Can be specified as a local register or a constant. If a register bit is specified for *Register* above (e.g. #DI3.1, #R1.1) then the *Number of Points* is the number of bits to send otherwise the *No. of Points* is the number of registers (or analog channels) to send. Note: when sending analog channels (#AO or #AI) every analog module is assigned 8 points and every digital module is assigned 16 points. A multi IO module with analog channels (such as the IO-4 or IO-3) is also assigned 8 points. Note: two points must be specified for each float or long that is sent.

Examples:	Register	Points
1. To send Chs 1-16 of #DI3 and #DI4	#DI3.1	32
2. To send Chs 1-8 of #AI6 and #AI7	#AI6.1	16
3. To send AI Chs 1-4 of IO-3 Modules 14 and 15	#AI14.2	12

Example 3 shown above will send #AI14.2, #AI14.3, #AI14.4, #AI14.5, #AI14.6 (not used), #AI14.7 (not used), #AI14.8 (not used), #AI15.1 (not used), #AI15.2, #AI15.3, #AI15.4 and #AI15.5. Alternatively, two TX Modbus blocks could be used that sent 4 points each.



Rx Modbus

Polls 16-bit registers or digital channels from a Modbus device. The data received from a Modbus device is stored in network registers corresponding to the address of the Modbus device.

Comment: A 12-character description.

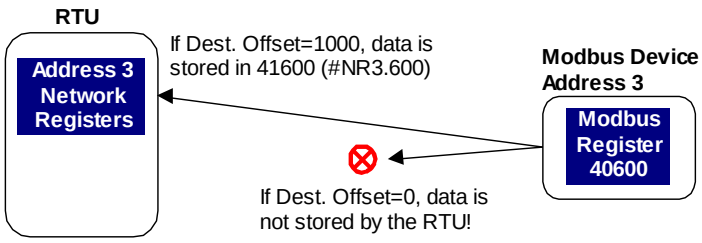
RTU #: (#R1 to #R2048 or 1 to 249) The source Modbus RTU (or PLC) to poll the data from. Can be specified as a local register or a constant.

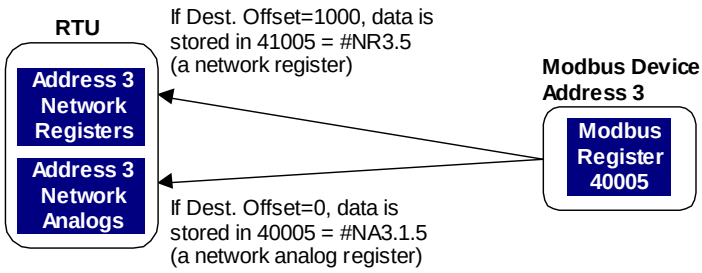
Dest. Offset: (-9999 to 9999 [or -1 to -65535 when using extended addressing]) When data is polled from a Modbus device it is stored in the RTU's network registers. An RTU is unable to store the complete Modbus register range. Modbus registers that are out of range can be moved into the correct range by using a Destination Offset. Modbus registers that are in range but would be stored in Network Analog or Network Digital registers can also be offset so that they are stored in the more easily accessible Network Registers. *Note: If a positive or zero Destination Offset is required, extended addressing cannot be used.*

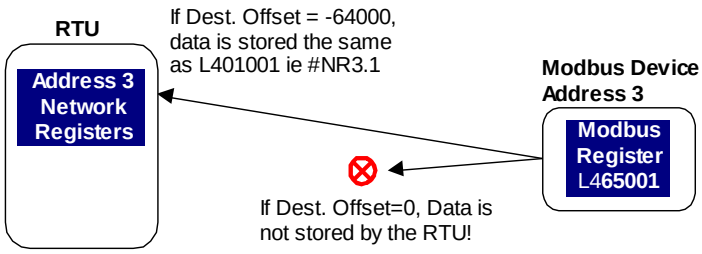
Modbus Address	Extended Modbus Address	Stored In RTU At *
00,001 to 01,024	L000,001 to L001,024	#NDrrr.1.1 to #NDrrr.64.16
01,025 to 02,000	L001,025 to L002,000	Not Stored!
02,001 to 09,999	L002,001 to L009,999	#NRrrr.1.1 to #NRrrr.500.15
	L010,000 to L065,535	Not Stored!
10,001 to 11,024	L100,001 to L101,024	Same as 00,001 to 01,024
11,025 to 12,000	L101,025 to L102,000	Same as 01,025 to 02,000
12,001 to 19,999	L102,001 to L109,999	Same as 02,001 to 09,999
	L110,000 to L165,535	Not Stored!
30,001 to 39,999	L300,001 to L309,999	Same as 40,001 to 49,999
40,001 to 40,512	L400,001 to L400,512	#NArrr.1.1 to #NArrr.64.8
40,513 to 41,000	L400,513 to L401,000	Not Stored!
41,001 to 43,048	L401,001 to L403,048	#NRrrr.1 to #NRrrr.2048
43,049 to 49,999	L403,049 to L409,999	Not Stored!
	L410,000 to L465,535	Not Stored!

* Where *rrr* is the source RTU (or PLC) address where the data came from.

Examples of how to use a Destination Offset are shown below.


RTU
Address 3 Network Registers
If Dest. Offset=1000, data is stored in 41600 (#NR3.600)
Modbus Device Address 3
Modbus Register 40600
If Dest. Offset=0, data is not stored by the RTU!


RTU
Address 3 Network Registers
Address 3 Network Analogs
If Dest. Offset=1000, data is stored in 41005 = #NR3.5 (a network register)
Modbus Device Address 3
Modbus Register 40005
If Dest. Offset=0, data is stored in 40005 = #NA3.1.5 (a network analog register)



MODBUS address: (1 to 49999 [or L000001 to L465535 when using extended addressing]) The starting address to request the registers or digital channels from in the source RTU (or PLC). The registers or channels are requested from consecutive addresses starting from the address specified here.

No. of Points: (#R1 to #R2048 or 1 to 123) The number of consecutive 16-bit registers or bits (channels) to poll. Can be specified as a local register or a constant. If a digital address is specified for *MODBUS address* above (00,000 or 10,000 range) then the *Number of Points* is the number of bits to poll otherwise the *No. of Points* is the number of registers (or analog channels) to poll. Note: two points must be specified for each float or long that is polled.

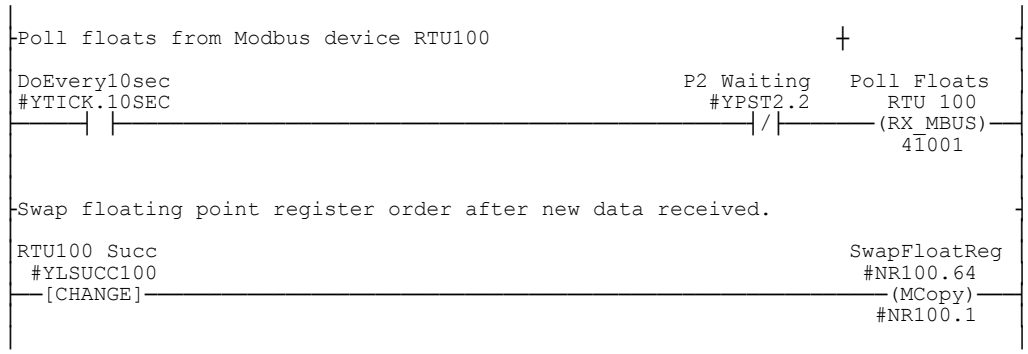
Communicating With A Modbus Device

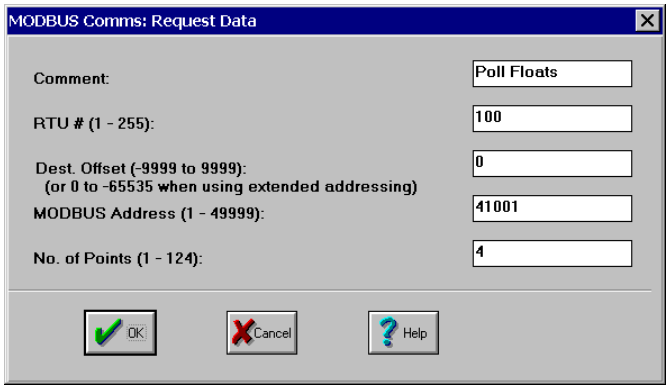
When using a Tx or Rx Modbus block or responding to Modbus messages, the RTU should be configured as follows:

- First ensure that the Modbus driver is supported by the type of RTU that is being used and that the latest firmware and driver are loaded in the RTU (as detailed in **protocols.pdf** available from http://helpdesk.servelec-semaphore.com).
- From **Configuration, Port List** set port *protocol* to one of the Modbus options eg. Mbus SCADA, S2. Configure the port for the correct *baudrate* (if using a serial port) or set the port **Type** to Ethernet (if using Ethernet). If using Ethernet, select the **Configure** button and configure the port's IP Address.
- If initiating Modbus messages from ladder logic, assign an address to the external device and add this address to the network list.
- Note 1: RTU address 174 should not be used as this corresponds to the Sync character at the start of Kingfisher Series 2 messages (AE). If a Modbus message is sent to RTU174, RTU174 will think it is a Kingfisher message.
- Note 2: When using an Allen Bradley Panelview Plus 1250 PLC to poll the RTU using Modbus over Ethernet: the Allen Bradley PLC is configured using KEPServer. Ensure the Device **ID** in KEPServer is set to XXX.XXX.XXX.XXX.YYY where XXX = RTU port Ethernet IP address and YYY = RTU Address (1-249). The Modbus **Port Number** to poll is 502.

Modbus Floating Point and Long Registers

Each float or long number is stored in two 16-bit registers. A Modbus device stores the two 16-bit registers in reverse order to a Kingfisher RTU (Kingfisher RTUs store the lower 16 bits in the lower register number). Before using floats or longs from a Modbus device or writing floats or longs to a Modbus device, the two 16-bit registers used for each float or long will need to be swapped as illustrated below.





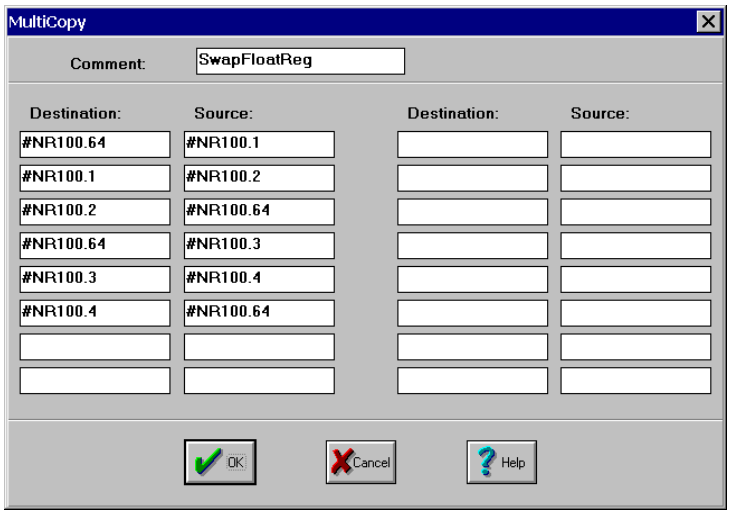


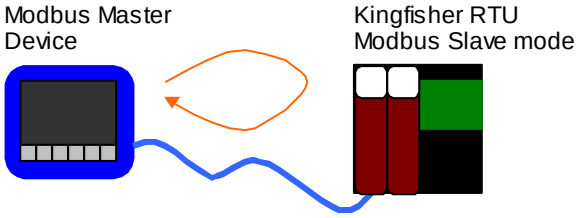
Figure: Example ladder logic used to poll 2 Modbus floating point numbers and then swap the 16-bit register order (note: #NR100.64 is used for temporary storage).

Modbus Address Mapping

The standard Modbus address ranges are as follows:

- Digital Outputs: 00,001 - 09,999 (Coils)
- Digital Inputs: 10,001 - 19,999 (Discrete Inputs)
- Analog Inputs: 30,001 - 39,999 (Input Registers)
- Analog Inputs/Outputs: 40,001 - 49,999 (Holding Registers)

Due to memory limitations, the RTU does not use the complete Modbus address range. The RTU responds to requests for Input Registers (30,001 - 39,999) as if they were Holding Registers (40,001 - 49,999 respectively). Local registers, analog and digital outputs can be read and written to as indicated below. The Modbus addresses that can be read from an RTU are listed below.



Can write Modbus addresses 41001 to 43048 to RTU
(Holding Registers 1001 to 3048 = #R1 to #R2048 respectively)
Can read Modbus addresses 40001 to 40512 and 41001 to 43048 from RTU.

Analog I/O (eg. AI-1/4/10, AO-2)
Note: IO-3/4 analog inputs start at Ch 2.
IO-3 analog output is channel 6.

Slot	Ch	Address
1	1-8	40,001 - 40,008
2	1-8	40,009 - 40,016
3	1-8	40,017 - 40,024
4	1-8	40,025 - 40,032
5	1-8	40,033 - 40,040
6	1-8	40,041 - 40,048
7	1-8	40,049 - 40,056
8	1-8	40,057 - 40,064
9	1-8	40,065 - 40,072
10	1-8	40,073 - 40,080
11	1-8	40,081 - 40,088
12	1-8	40,089 - 40,096
13	1-8	40,097 - 40,104
14	1-8	40,105 - 40,112
15	1-8	40,113 - 40,120
16	1-8	40,121 - 40,128
..		40,000 + [(Slot-1)x8] + Ch
64	1-8	40,505 - 40,512

Register Read/Write * (eg. #R1)

Reg.	Address
1	41,001
2	41,002
3	41,003
4	41,004
5	41,005
6	41,006
7	41,007
8	41,008
9	41,009
10	41,010
11	41,011
12	41,012
13	41,013
14	41,014
15	41,015
16	41,016
..	41,000 + Reg.
2048	43,048

* When used with a bit mask, these addresses can also be used for bit read/writes.

Digital Input (eg. DI-10 Channel 1)		
Slot	Ch	Address
1	1-16	10,001 - 10,016
2	1-16	10,017 - 10,032
3	1-16	10,033 - 10,048
4	1-16	10,049 - 10,064
5	1-16	10,065 - 10,080
6	1-16	10,081 - 10,096
7	1-16	10,097 - 10,112
8	1-16	10,113 - 10,128
9	1-16	10,129 - 10,144
10	1-16	10,145 - 10,160
11	1-16	10,161 - 10,176
12	1-16	10,177 - 10,192
13	1-16	10,193 - 10,208
14	1-16	10,209 - 10,224
15	1-16	10,225 - 10,240
16	1-16	10,241 - 10,256
..		10,000 + [(Slot-1)x16] + Ch
64	1-16	11,009 - 11,024

Register Bit Read * (eg. #R1.1)		
Reg.	Bit	Address
1	1-16	12,001 - 12,016
2	1-16	12,017 - 12,032
3	1-16	12,033 - 12,048
4	1-16	12,049 - 12,064
5	1-16	12,065 - 12,080
6	1-16	12,081 - 12,096
7	1-16	12,097 - 12,112
8	1-16	12,113 - 12,128
9	1-16	12,129 - 12,144
10	1-16	12,145 - 12,160
11	1-16	12,161 - 12,176
12	1-16	12,177 - 12,192
13	1-16	12,193 - 12,208
14	1-16	12,209 - 12,224
15	1-16	12,225 - 12,240
16	1-16	12,241 - 12,256
..		12,000 + [(Reg. -1)x16] + Bit
500	1-**15**	19,985 - 19,999

Digital Output (eg. DO-2 Channel 1) Note: IO-2/3/4 digital outputs start at Ch 9		
Slot	Ch	Address
1	1-16	00,001 - 00,016
2	1-16	00,017 - 00,032
3	1-16	00,033 - 00,048
4	1-16	00,049 - 00,064
5	1-16	00,065 - 00,080
6	1-16	00,081 - 00,096
7	1-16	00,097 - 00,112
8	1-16	00,113 - 00,128
9	1-16	00,129 - 00,144
10	1-16	00,145 - 00,160
11	1-16	00,161 - 00,176
12	1-16	00,177 - 00,192
13	1-16	00,193 - 00,208
14	1-16	00,209 - 00,224
15	1-16	00,225 - 00,240
16	1-16	00,241 - 00,256
..		(Slot -1)x16 + Ch
64	1-16	01,009 - 01,024

Register Bit Read/Write * (eg. #R1.1)		
Reg.	Ch	Address
1	1-16	02,001 - 02,016
2	1-16	02,017 - 02,032
3	1-16	02,033 - 02,048
4	1-16	02,049 - 02,064
5	1-16	02,065 - 02,080
6	1-16	02,081 - 02,096
7	1-16	02,097 - 02,112
8	1-16	02,113 - 02,128
9	1-16	02,129 - 02,144
10	1-16	02,145 - 02,160
11	1-16	02,161 - 02,176
12	1-16	02,177 - 02,192
13	1-16	02,193 - 02,208
14	1-16	02,209 - 02,224
15	1-16	02,225 - 02,240
16	1-16	02,241 - 02,256
..		2,000+[(Reg.-1)x16]+Ch
500	1-**15**	09,985 - 09,999

* It is only possible to access up to Ch15 of Register 500 (corresponding to address 09,999 or 19,999) as a register bit. Ch16 of R500 (and after) cannot be accessed as a digital input or output. However, it is possible to read and write to all of the local registers using integer values.

Driver - Trio E-Series Radio

Rx Trio E-Series Radio

Reads status or statistical information from a Trio E-Series radio.

Comment: A 12-character description.

RTU # (1-255): Address assigned to the Trio radio. Data returned from the Trio radio is stored in network registers corresponding to this address.

Radio ID: (0 or long constant) When set to 0, communicates with the local radio. Otherwise set to the serial number of the target radio. Specifying 0 will allow the local radio to be replaced without having to update the RTU configuration each time.

Command: (Read Status Data or Read Statistics Data) Returns data as follows (where #Rx is the Destination Register defined below):

Read Status Data (returns eight 16-bit signed registers)

Data Description	Register	Units	Example
Temperature	#Rx	0.1 Deg C	245 = 24.5 deg C
Signal Indicator (RSSI)	#Rx+1	0.1 dBm	-701 = -70.1dBm
Fwd Tx Power (Remote) *	#Rx+2	0.1 dBm	204 = 20.4 dBm
Supply Volts	#Rx+3	0.1 V	138 = 13.8 V
Rx Freq Error	#Rx+4	0.1 Hz	254 = 25.4 Hz
Rev Tx Power (Remote) *	#Rx+5	0.1 dBm	77 = 7.7 dBm
Fwd Tx Power (Base Station) *	#Rx+6	0.1 dBm	126 = 12.6 dBm
Rev Tx Power (Base Station) *	#Rx+7	0.1 dBm	34 = 3.4 dBm

* If connected to a base station radio, the Fwd and Rev Tx Power (Remote) fields will be set to 0. Similarly, if connected to a remote radio, the Fwd & Rev Tx Power (Base Station) fields will be set to 0.

Read Statistics Data (returns four 32-bit long registers)

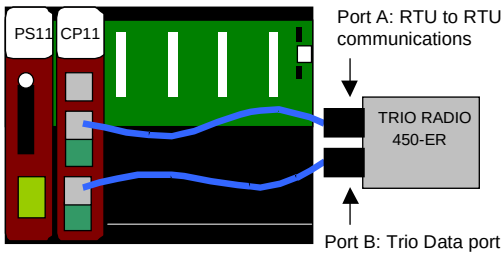
Data Description	Register
Bad Frame Count	#Lx
Good Frame Count	#Lx+2
Lost Sync Count	#Lx+4
Lost RSSI Count	#Lx+6

Destination Register: (#R1 to #R2048) Specified as a local register but indicates the first network register to begin storing the data from. When the Read Statistics Data command is being used, an odd numbered local register should be specified (#R1, #R3…#R2047) to allow the received data (in Long format) to be correctly displayed using Toolbox.

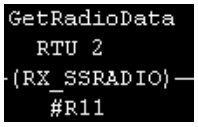
Communicating With A Trio E-Series Radio

When using an Rx Trio_E block, the RTU should be configured as follows:

- First ensure that the Trio E-Series Radio driver is supported by the type of RTU that is being used and that the latest firmware and driver are loaded in the RTU (as detailed in **protocols.pdf** available from http://helpdesk.servelec-semaphore.com).
- From **Configuration, Port List** configure the port for the correct *baudrate* (Trio radio default is 19200) and set the port *protocol* to TRIO Eseries.
- Assign an address to the Trio radio and add this address to the network list in **Configuration, Network List** (this must be a unique address in the RTU network).
- Configure an RX TRIO ladder block to poll the local or remote Trio radio. To poll the local radio set *Radio ID* to 0. To poll a remote Trio radio set *Radio ID* to the serial number of the remote radio.
- Both ports of the local Trio radio must be used if the radio is also used for RTU to RTU communications. Trio radio port A is used for RTU to RTU communications while Port B allows the radio data to be polled.



Driver - Spread Spectrum Radio



Rx Spread Spectrum Radio

Reads diagnostic information from a MaxStream XTend spread spectrum radio.

Comment: A 12-character description.

RTU # (1-255): Address assigned to the spread spectrum radio. Data returned from the radio is stored in network registers corresponding to this address.

Destination Register: (#R1 to #R2045) Specified as a local register but indicates the first network register to begin storing radio data from. The following data is returned (where #Rx is the configured Destination Register):

Data Description	Register	Range	Units	Example
Board voltage	#Rx	280-575	0.01 Volts	511 = 5.11 Volts
Signal strength of last received message packet	#Rx+1	110 to 40 or 32768*	-dBm	63 = -63 dBm 32768 = Undetermined
Number of receive errors (message packets)	#Rx+2	0 to 65535	Errors	
Number of valid received message packets	#Rx+3	0 to 65535	Successes	

* Note: Signal strength returns 32768 (undetermined) when no message packets have been received since the radio was powered up.

Communicating With A Spread Spectrum Radio

When using an Rx SS RADIO ladder block, the RTU should be configured as follows:

- First ensure that the Spread Spectrum Radio driver is supported by the type of RTU that is being used and that the latest firmware and driver are loaded in the RTU (as detailed in protocols.pdf available from http://helpdesk.servelec-semaphore.com)
- From **Configuration, Port List** set the port *Type* to SS_RADIO and *Baud Rate* to 9600 (for 9XTend US / Australian radio) or 19200 (for 24XStream International radio). *Pre Tx* and *Post Tx* can be left set to 0. *Protocol* can also be left set to Series 2. Note: port *Type* can be set to RS232 if the radio does not need to be configured with an initialisation string. When using the SS_RADIO port Type, an initialisation string can be configured by selecting the **Configure** button. Please ensure that the settings used for *Vendor ID*, *Destination Address* and *Hopping Pattern* are the same for all radios in the network.
- From **Configuration, Network List** assign an address to the spread spectrum radio (this should be a unique address in the RTU network). The network registers of this address will be used to store the spread spectrum radio data. Set *Port #* to the spread spectrum radio port number and Timeout to 4000 ms.
- When communicating between RTUs using spread spectrum radios, the *Timeout* of the network link to the remote RTU should be set to 2000 ms.
- Configure an RX_SSRADIO ladder block to poll the local SS radio. An example is shown below.

Reading Radio Data

The example below shows how radio data can be obtained after a successful message is received from RTU2 or RTU3. The radio has been assigned address 100. RTU2 radio data is stored from RTU100 network register 1 (Destination Register = #R1). RTU3 radio data is stored from network register 5 (Destination Register = #R5).

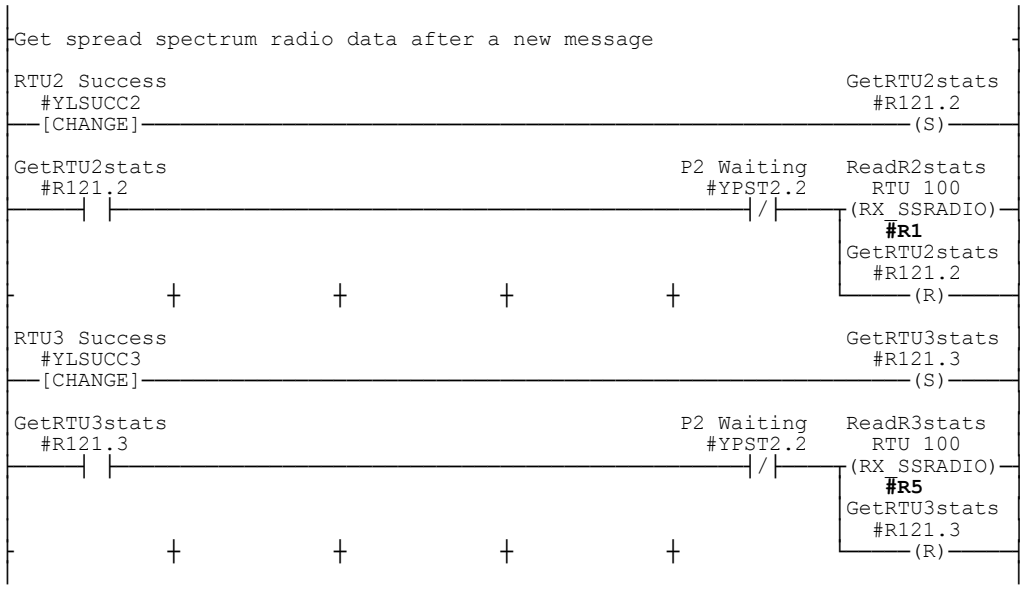


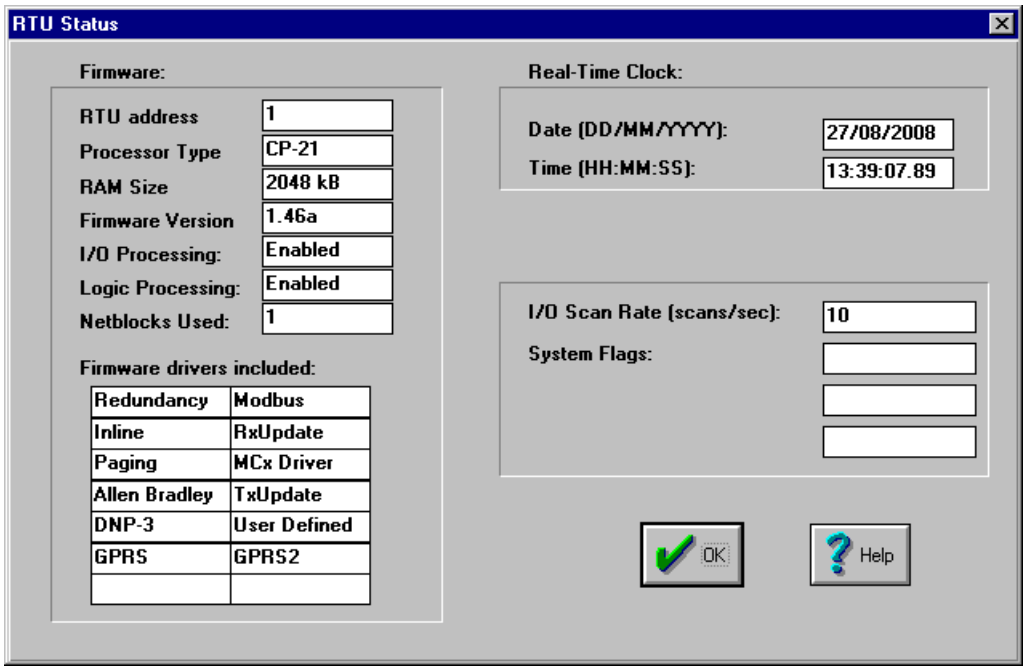
Figure: Example ladder logic used to read radio data for RTUs 2 and 3 (assumes spread spectrum radio is connected to port 2)

8. View Menu

RTU Status	Displays operational state of the RTU
Hardware Overview	Displays modules detected on the backplane(s) and the data they contain
Local Registers	Displays contents of local registers
Network Registers	Displays the local, digital or analog registers received from a remote RTU
Timer Registers	Displays contents of timer registers
Freeform Display	Allows pages of registers to be defined in a text file and then displayed live
Event Logging	Displays the event logs in the RTU
RTU Comms Statistics	Displays communication successes and fails for each remote RTU
PC Comms Statistics	Displays communication successes and fails for all messages sent by Toolbox
Read Drivers Info	Displays information about the firmware drivers loaded in the RTU
Auto Detect	Detects the address and baud rate of the local RTU

View - RTU Status

Displays the operational state of the RTU.



RTU Address: (1 to 249) Address of the RTU that Toolbox is communicating with.

Processor Type: PC-1, CP-1, CP-10/11, CP-21, SBX, ERS Micro, LP-1/2/3, SB-1 or Micro-4.

RAM Size: Battery backed RAM size.

Firmware Version: Version of the operating code as stored in flash memory.

I/O Processing: Enabled or Disabled. Status of IO module scanning.

Ladder Processing: Enabled or Disabled. Status of ladder logic processing.

Netblocks Used: The number of network register blocks that have been used. The total number of network blocks that are available can be checked by selecting the Check memory usage button in **Configuration, Memory**. If the maximum limit is exceeded a *Netblks Overrun* system flag is displayed.

Date, Time: Real time clock settings in the RTU. The displayed time is updated every time Toolbox polls the RTU. The polling rate is determined by the Comms Repeat Rate setting in Configuration, PC Setup.

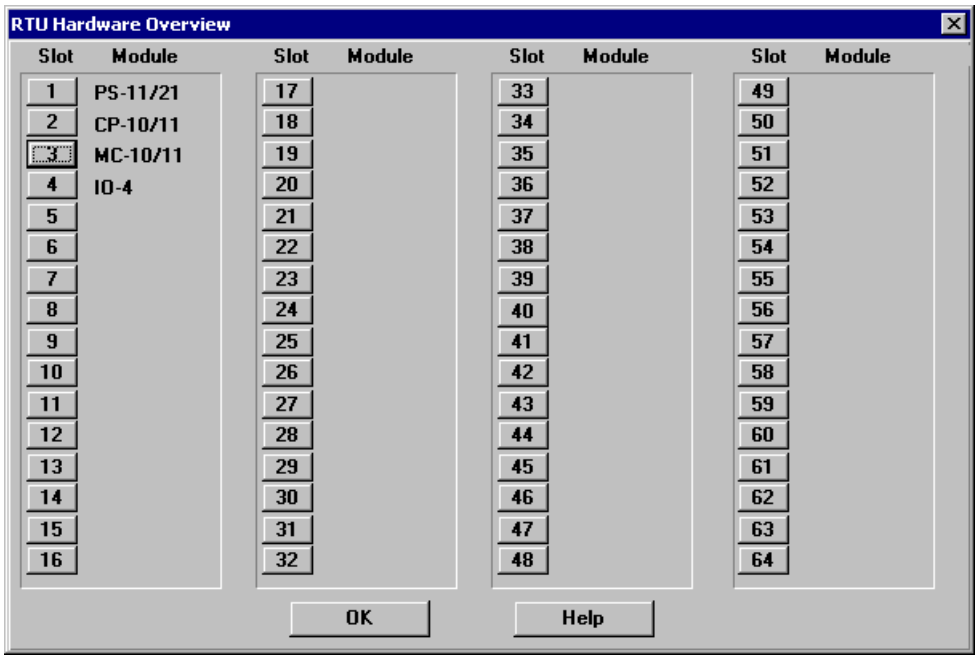
I/O Scan Rate (scans/sec): Rate of scanning the IO modules and processing ladder logic (if enabled).

System Flags: General RTU operation flags (from the system register #YFLAGS). Note: the flags *Scan Overrun* and *Log Overrun* are status indications only.

Firmware drivers included: The names of firmware drivers and special functions that have been successfully loaded in the RTU. Some drivers are included in firmware (eg. Redundancy). CP-21 has most drivers included in firmware.

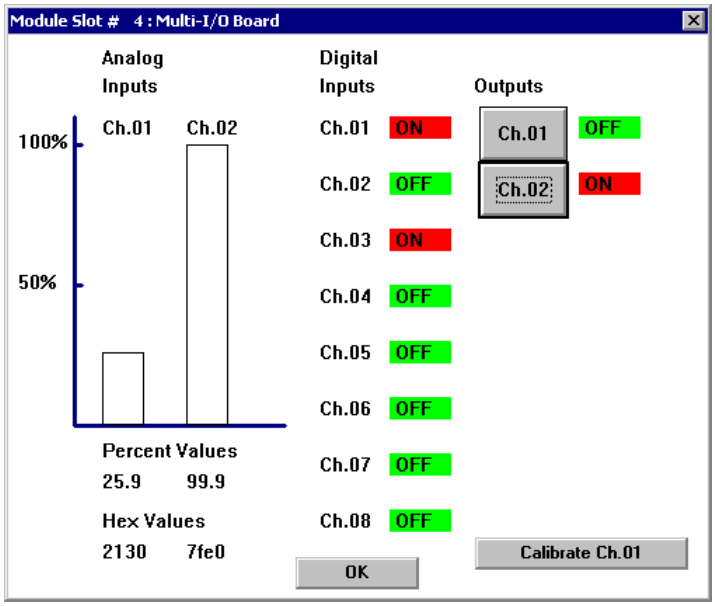
View - Hardware Overview

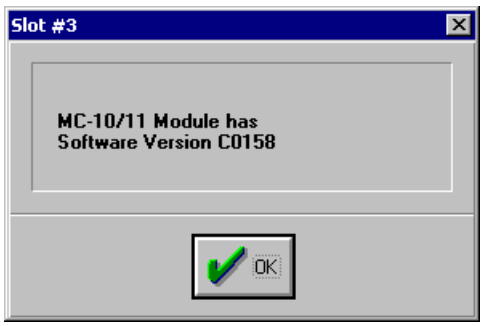
Displays the modules detected on the backplane(s) and the data they contain. When an MC module is selected, displays the MC firmware version.



If the modules have appeared in the wrong slot positions, this can be changed by powering down the RTU and setting the DIP switches on the backplane to the correct rack number (as detailed on the backplane).

Module details can be viewed by selecting the **Slot** button alongside the module (**1** to **64**).





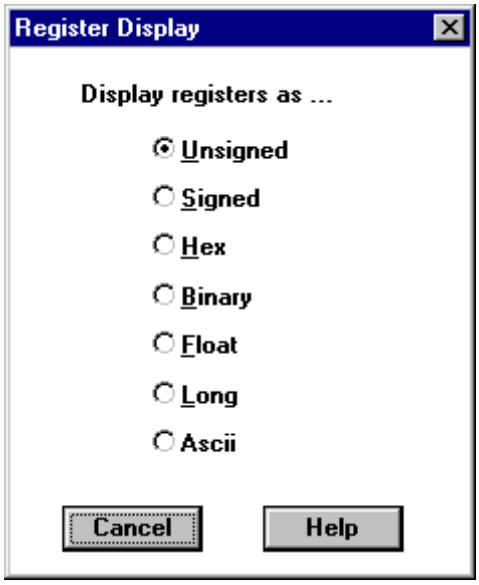
Note: Analog or digital outputs can be manually set from the hardware overview provided the outputs are not being controlled by ladder logic. To manually set outputs without the RTU overwriting them, first disable ladder logic processing using **Utilities, Enable/Disable, Disable Logic Processing**.

View - Local Registers

Displays contents of local registers.

All the local registers (1 to 2048) can be viewed by selecting the **PageUp** or **PageDown** buttons or by jumping to the relevant register by selecting the **GotoReg** button. Any local register can also be written to by selecting the register button and entering a value.

Display As Allows the default display format to be changed as detailed below. Note: selecting **Hex** or **Binary** will display registers in hexadecimal format when viewing ladder logic in Debug mode.



Unsigned: (0 to 65535) Raw 16-bit integer numbers.

Signed: (-32768 to +32767) Signed 15-bit numbers. If channel 16 is ON, the number is negative.

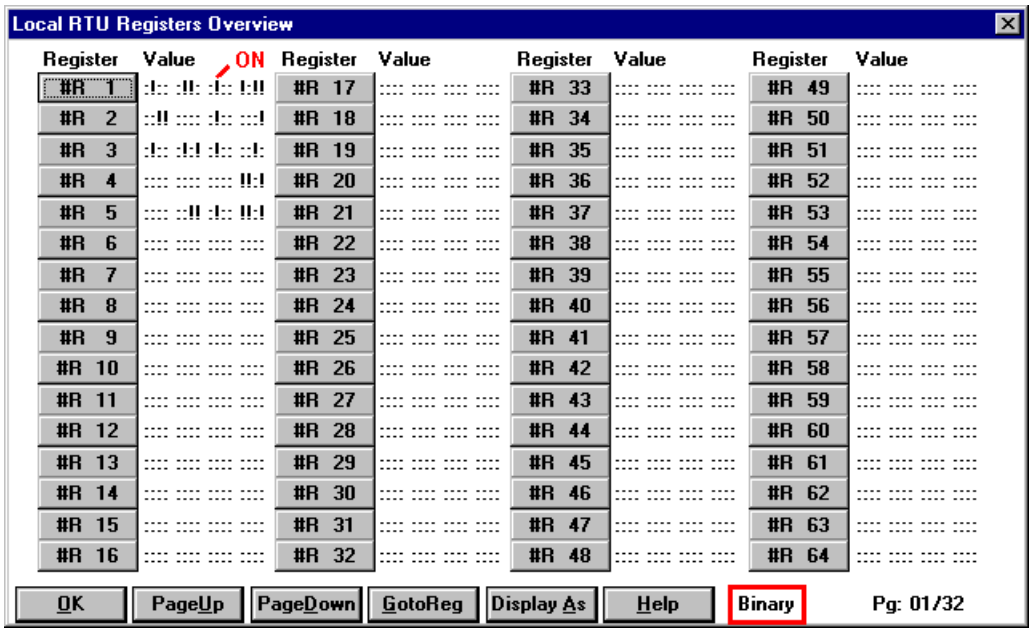
Hex: (0 to FFFF Hex) Hexadecimal numbers.

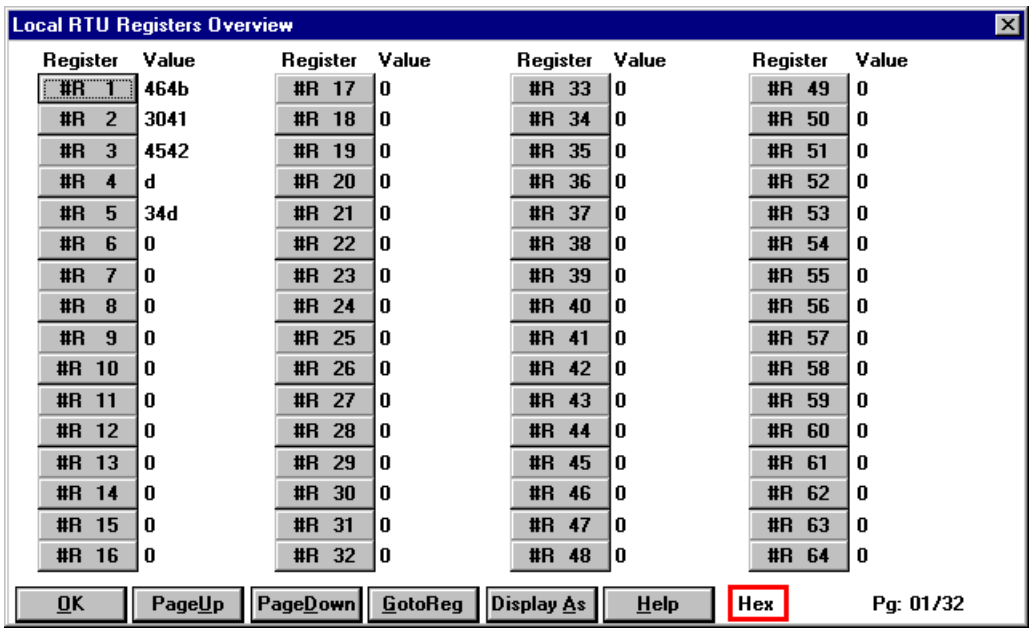
Binary: (**!** = ON, **:** = OFF) Displays the state of each bit. Right-most bit (LSB) is channel 1, left-most bit (MSB) is channel 16.

Float: ($3.4x10^{-38}$ to $3.4x10^{38}$) Signed 32-bit numbers. To be able to store numbers of this size, the RTU stores floating point numbers in 2 consecutive local registers and so only odd numbered float registers are displayed. Eg. #F5 uses #R5 and #R6 while #F7 uses #R7 and #R8. Float numbers can be entered as a signed decimal number or by using the exponential format *yX.XXXXeyZZ* where *X.XXXX* is the number, *y* is the sign (+ or -) and *ZZ* is the exponential power.

Long: (-2,147,483,648 to 2,147,483,647) Signed 32-bit numbers. Like float numbers, long numbers use two consecutive local registers and so only odd numbered long registers are displayed. Eg. #L5 uses #R5 and #R6 while #L7 uses #R7 and #R8.

ASCII: Displays ASCII characters (A-Z, a-z, 0-9, ( ) + - < > etc). Allows strings to be viewed that have been copied to local registers using the String Copy block. Two 8-bit ASCII characters can be stored in one local register. The high byte (Channels 9-16) is displayed as the *right* character and the low byte (channels 1-8) is displayed as the *left* character. Eg. if #R1 contains 464B Hex, this will be displayed as KF (K=4B Hex and F=46 Hex).


Local RTU Registers Overview
Register Value ON Register Value Register Value Register Value
#R 1 :!:: :!!: :!:: !:!! #R 17 :::: :::: :::: :::: #R 33 :::: :::: :::: :::: #R 49 :::: :::: :::: ::::
#R 2 ::!! :::: :!:: :::! #R 18 :::: :::: :::: :::: #R 34 :::: :::: :::: :::: #R 50 :::: :::: :::: ::::
#R 3 :!:: :!:! :!:: ::!: #R 19 :::: :::: :::: :::: #R 35 :::: :::: :::: :::: #R 51 :::: :::: :::: ::::
#R 4 :::: :::: :::: !!:! #R 20 :::: :::: :::: :::: #R 36 :::: :::: :::: :::: #R 52 :::: :::: :::: ::::
#R 5 :::: ::!! :!:: !!:! #R 21 :::: :::: :::: :::: #R 37 :::: :::: :::: :::: #R 53 :::: :::: :::: ::::
#R 6 :::: :::: :::: :::: #R 22 :::: :::: :::: :::: #R 38 :::: :::: :::: :::: #R 54 :::: :::: :::: ::::
#R 7 :::: :::: :::: :::: #R 23 :::: :::: :::: :::: #R 39 :::: :::: :::: :::: #R 55 :::: :::: :::: ::::
#R 8 :::: :::: :::: :::: #R 24 :::: :::: :::: :::: #R 40 :::: :::: :::: :::: #R 56 :::: :::: :::: ::::
#R 9 :::: :::: :::: :::: #R 25 :::: :::: :::: :::: #R 41 :::: :::: :::: :::: #R 57 :::: :::: :::: ::::
#R 10 :::: :::: :::: :::: #R 26 :::: :::: :::: :::: #R 42 :::: :::: :::: :::: #R 58 :::: :::: :::: ::::
#R 11 :::: :::: :::: :::: #R 27 :::: :::: :::: :::: #R 43 :::: :::: :::: :::: #R 59 :::: :::: :::: ::::
#R 12 :::: :::: :::: :::: #R 28 :::: :::: :::: :::: #R 44 :::: :::: :::: :::: #R 60 :::: :::: :::: ::::
#R 13 :::: :::: :::: :::: #R 29 :::: :::: :::: :::: #R 45 :::: :::: :::: :::: #R 61 :::: :::: :::: ::::
#R 14 :::: :::: :::: :::: #R 30 :::: :::: :::: :::: #R 46 :::: :::: :::: :::: #R 62 :::: :::: :::: ::::
#R 15 :::: :::: :::: :::: #R 31 :::: :::: :::: :::: #R 47 :::: :::: :::: :::: #R 63 :::: :::: :::: ::::
#R 16 :::: :::: :::: :::: #R 32 :::: :::: :::: :::: #R 48 :::: :::: :::: :::: #R 64 :::: :::: :::: ::::
OK
PageUp
PageDown
GotoReg
Display As
Help
Binary
Pg: 01/32

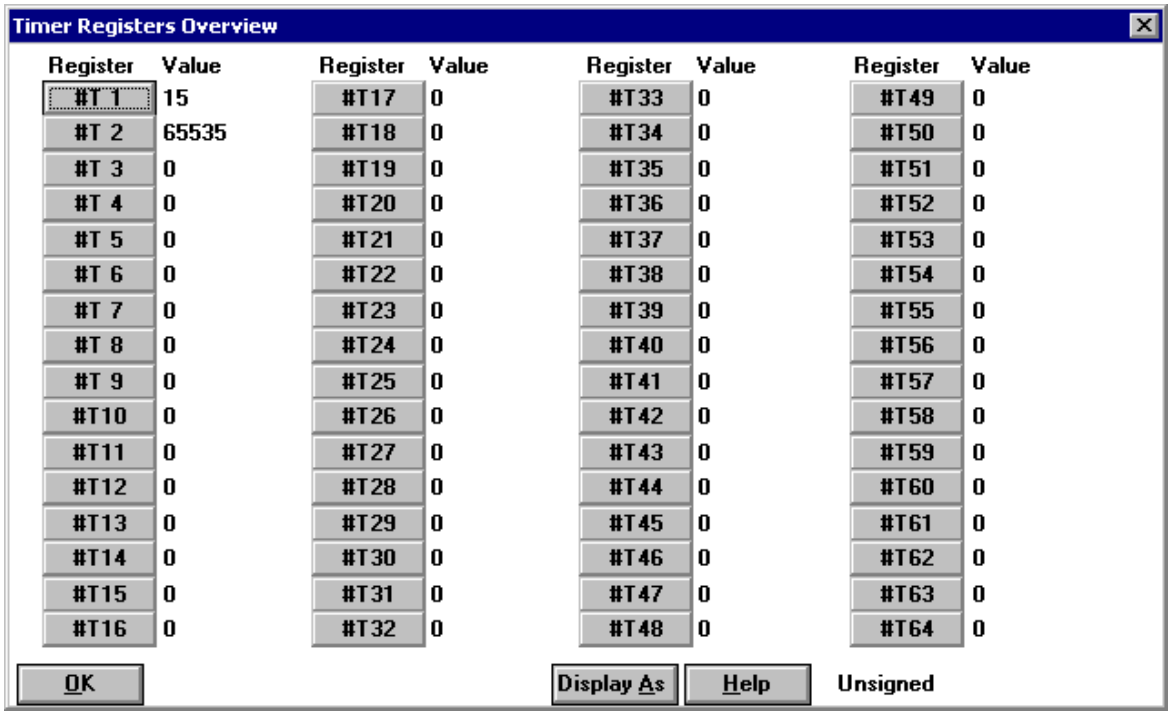


View - Timer Registers

Displays contents of the 64 timer registers.

These timers can be written to by selecting the timer button (eg. select #**T 1**) and entering a value. If ladder logic is using a timer register, the timer register's value can still be manually changed.

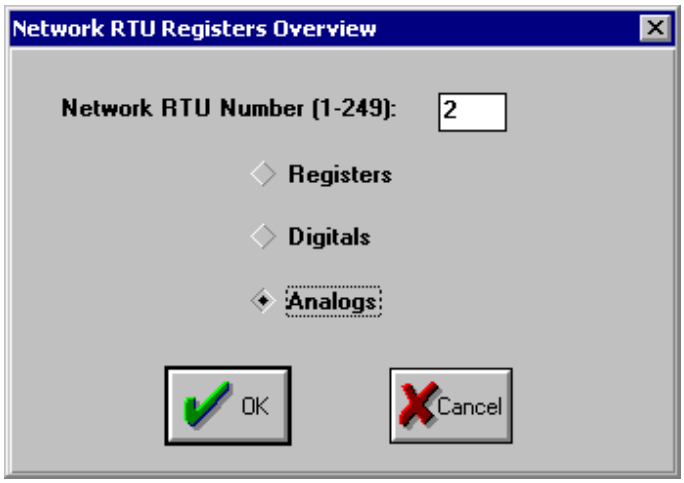
Display As The various display formats are detailed in View - Local Registers.



View - Network Registers

Displays the local, digital or analog registers received from a remote RTU.

Display As The various display formats are detailed in View - Local Registers.



Network RTU Number (1-249): The address of the remote RTU to view data from. Note: entering 0 will display the registers of the local RTU.

Registers: When selected, displays the local registers received from the remote RTU.

Digitals: When selected, displays the digital registers received from the remote RTU. Note: the digital register is the first of eight hardware registers received from each IO module.

Analogs: When selected, displays the eight hardware registers received from each IO module in the remote RU. The first hardware register (#NA xx.1) contains the digital channels if the IO module has digital inputs or outputs.

Network Registers Overview, RTU #2

Register	Value	Register	Value	Register	Value	Register	Value
#NA 1.1	0	#NA 3.1	6	#NA 5.1	0	#NA 7.1	0
#NA 1.2	0	#NA 3.2	12800	#NA 5.2	0	#NA 7.2	0
#NA 1.3	0	#NA 3.3	16128	#NA 5.3	0	#NA 7.3	0
#NA 1.4	0	#NA 3.4	0	#NA 5.4	0	#NA 7.4	0
#NA 1.5	0	#NA 3.5	11520	#NA 5.5	0	#NA 7.5	0
#NA 1.6	0	#NA 3.6	0	#NA 5.6	0	#NA 7.6	0
#NA 1.7	0	#NA 3.7	0	#NA 5.7	0	#NA 7.7	0
#NA 1.8	0	#NA 3.8	0	#NA 5.8	0	#NA 7.8	0
#NA 2.1	0	#NA 4.1	0	#NA 6.1	0	#NA 8.1	0
#NA 2.2	0	#NA 4.2	0	#NA 6.2	0	#NA 8.2	0
#NA 2.3	0	#NA 4.3	0	#NA 6.3	0	#NA 8.3	0
#NA 2.4	0	#NA 4.4	0	#NA 6.4	0	#NA 8.4	0
#NA 2.5	0	#NA 4.5	0	#NA 6.5	0	#NA 8.5	0
#NA 2.6	0	#NA 4.6	0	#NA 6.6	0	#NA 8.6	0
#NA 2.7	0	#NA 4.7	0	#NA 6.7	0	#NA 8.7	0
#NA 2.8	0	#NA 4.8	0	#NA 6.8	0	#NA 8.8	0

OK | PageUp | PageDown | GotoReg | Display As | Help | Unsigned | Pg: 1/8

Figure: Network Analogs For RTU2 (only slot 3 has analog inputs)

View - Freeform Display

Allows pages of variables (registers) to be defined in a text file and then displayed live.

- Variables are updated each time Toolbox polls the RTU
- Variables are referred to on the text page by using **%n** where n=1 to 999
- Each variable must be defined at the end of the text file under the heading **%VARIABLES** (format for defining variables is detailed below)
- A variable can only be referred to once on any display page
- Up to 100 variables can be defined on each page
- When Freeform Display runs, variable references (%n) and subsequent characters are overwritten by the variable's value. Space characters must be added after a variable reference to allow the value to be displayed correctly.
- Any data address (#etc) or a Tagname can be used for a variable. When using Tagnames, Tagnames must be defined in the Variables List of an RTU site, the site must be in an open project and the site must be selected before Freeform Display is started.
- Analog values can be displayed in signed engineering units (assumes a raw range of 32767). Eg. 0 to 32760 can be displayed as -20.0 to 70.0 degC.
- Multiple pages of text and variables can be displayed. Press **PgUp** or **PgDn** to change pages.
- Page breaks are specified in the text file using **%NEWPAGE**

Example Freeform Text File

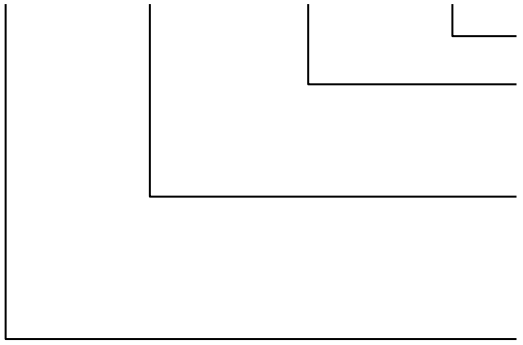
(The following text can be copied from Toolbox help then pasted and saved using Microsoft **Notepad**)

```
------ FREEFORM DISPLAY EXAMPLE --------

Register #R1 unsigned, right justified:         %1
Register #R1 Channel 1 status:                  %2
Floating Point Register #F3 with 2 dec. places: %3
Long Register #L5:                              %4
"TestLabel" from Variables List status:         %5

PgDn for page 2

%NEWPAGE

PgUp for page 1

RTU address showing 3 digits zero padded:       %6
Firmware version in hexadecimal format:         %7
LP-1/2/3 Temperature in engineering units:      %8     degC

%VARIABLES
1 #R1         %5u
2 #R1.1       %b "OFF" "ON "
3 #F3         %8.2f
4 #L5         %i
5 TestLabel   %b   "OK    " "FAULT"
6 #YADDRESS   %03u
7 #YFIRMW     %-4X
8 #AI1.7      %3.1s -23 77
```

Freeform Variable Definitions

Variables are defined in the text file after the heading **%VARIABLES** using the following format:

```
%VARIABLES
1      #R2.1        %b     "OFF" "ON "
```



Additional text settings

Display format: many options are available as detailed below.
Display format must start with a **%**

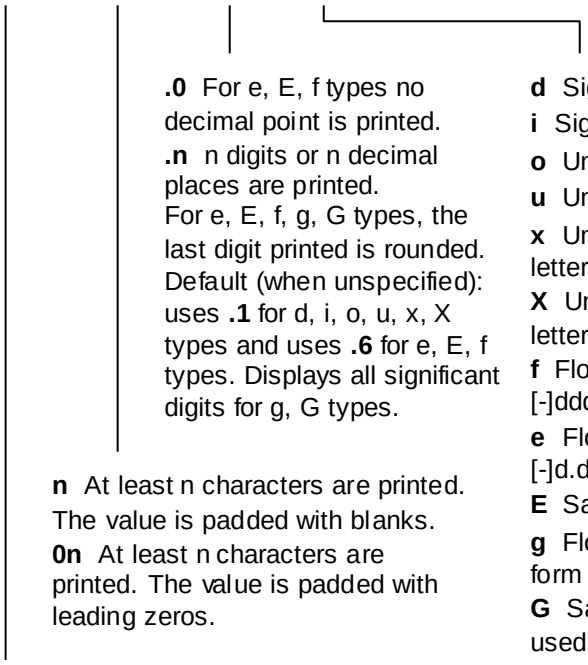
Address or tagname: Any valid data address starting with a **#** (eg. #R1, #YDIAG.1) or the TagName of a variable in the Variables List of the selected RTU site in an open project.

Variable number: the variable number (1 to 999) used on the text page as **%1 to %999**

Display Formats

%[**-**/**+**/#][n/0n][.0/.n]type

Note: bracketed [ ] parameters are optional. n=1 to 15.



.0 For e, E, f types no decimal point is printed.
.n n digits or n decimal places are printed.
For e, E, f, g, G types, the last digit printed is rounded.
Default (when unspecified): uses **.1** for d, i, o, u, x, X types and uses **.6** for e, E, f types. Displays all significant digits for g, G types.

n At least n characters are printed. The value is padded with blanks.
0n At least n characters are printed. The value is padded with leading zeros.

- Left justified

+ Value starts with a + or -

If type is o, value begins with a 0. If type is x or X, value starts with 0x. If type is e, E or f, value will have a decimal point. If type is g or G, value will have a decimal point and trailing zeros will not be removed. Default (when not specified): If negative, value starts with **-**

d Signed decimal integer

i Signed decimal integer

o Unsigned octal integer

u Unsigned decimal integer

x Unsigned hexadecimal integer using lower-case letters a-f

X Unsigned hexadecimal integer using capital letters A-F

f Floating point signed value of the form [-]dddd.dddd

e Floating point signed value of the form [-]d.dddd or e[+/-]ddd

E Same as e, but with capital E for exponent

g Floating point signed value in either the f or e form based on given value and precision

G Same as g, but with E for exponent if e format is used

b Bit string. Displays the first string if the bit is False or displays the second string if the bit is True. Strings must be separated by tabs or spaces.

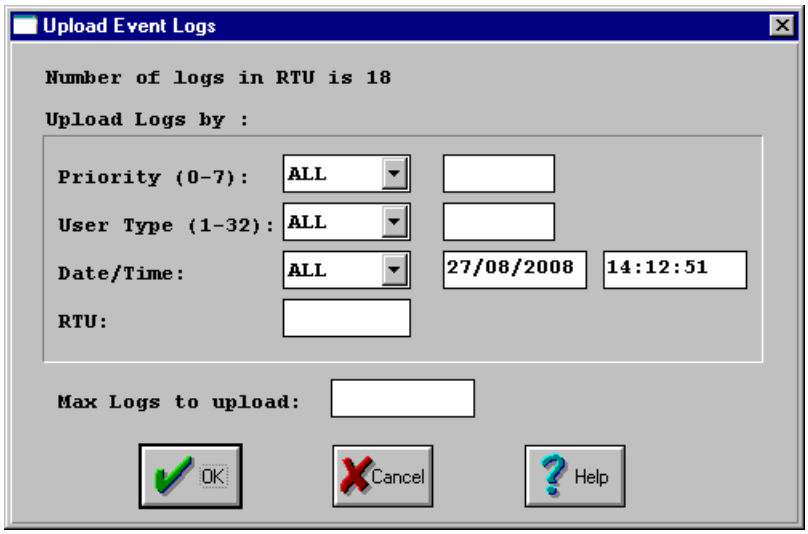
a Animated string. If the value of the variable is 0, the first string is printed. If the value is 1 the second string is printed, etc. If the value exceeds the number of given strings the last string is printed. Strings must be separated by tabs or spaces.

s Scaled value. Displays a signed scaled value assuming the raw range of the variable is 32767. The signed engineering limits corresponding to a raw input of 0 and 32767 respectively must be specified as additional strings separated by tabs or spaces.

View - Event Logging

Displays the event logs in the RTU.

If there are no logs in the RTU, the message “No Logs In RTU!” will be displayed otherwise the window shown below will be displayed.



Only the logs corresponding to the Priority, User Type, Date/Time and RTU settings will be uploaded. The maximum number of event logs to upload can also be specified (1-32760).

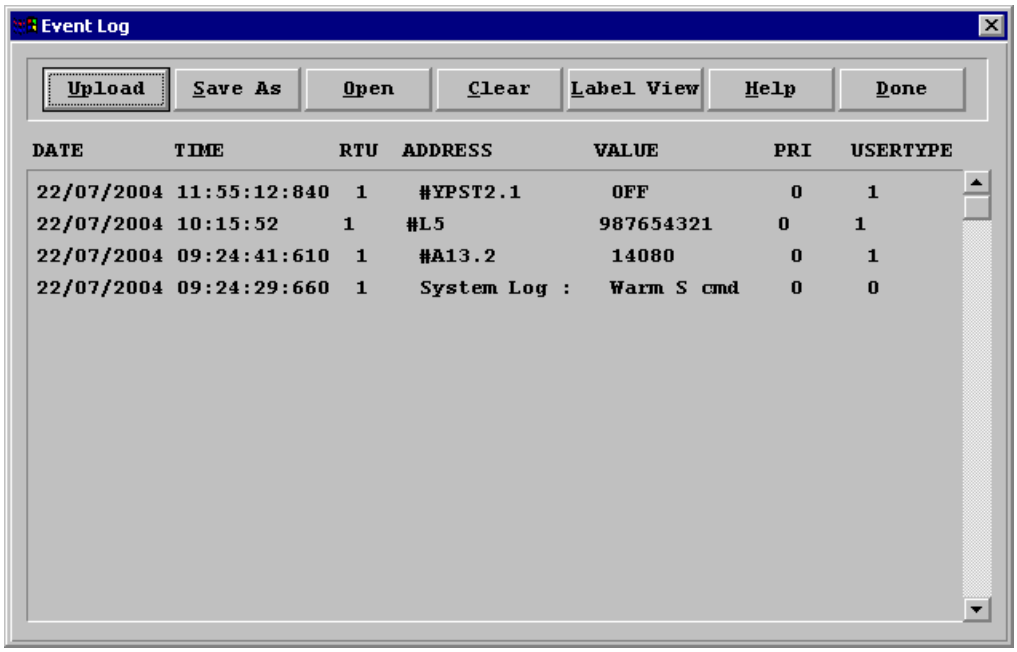


Figure: Example Event Log Window

Event Log window buttons:

- **Upload:** Upload specified event logs from the RTU.
- **Save As:** Save the uploaded logs in a DBF file (standard database format).
- **Open:** (To be implemented) Open an existing DBF log file for viewing.
- **Clear:** Clear all the event logs in the RTU.
- **Label View**: Displays register addresses as labels if the register address is defined in the Variables List of an RTU site in the currently opened project.
- **Done:** Close the Event Log window

View - RTU Comms Statistics

Displays communication successes and fails for each remote RTU that the local RTU has communicated with. Totals are reset when the RTU is warm started.

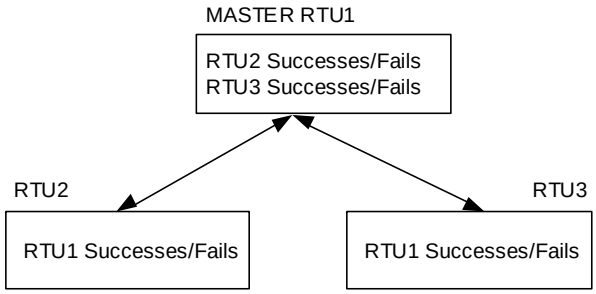


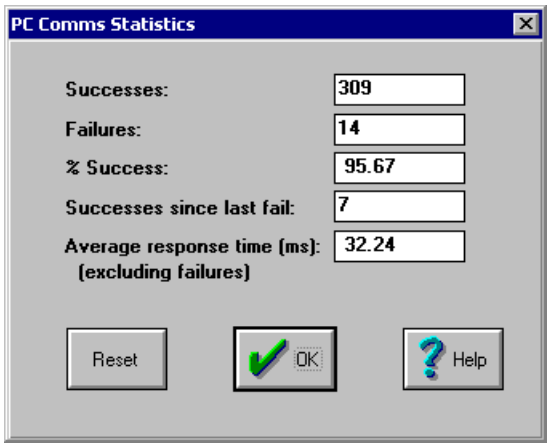
Figure: RTU Comms Statistics Stored In A Small Network

A success is recorded when a valid message is received or when a reply is received to a message sent to an RTU. A fail is recorded each time there is no reply to a message *attempt* within the timeout period. An example of Comms Statistics logged by RTU1 for communications to RTU2 are shown below.

Note: Fails will be recorded if too many messages are initiated (eg. each scan of ladder logic) or if the RTU does not wait long enough for a reply. To ensure the RTU waits until the message attempt has timed out, please use the #YPSTnn.2 bit as detailed in the topic Example - Polling Data.

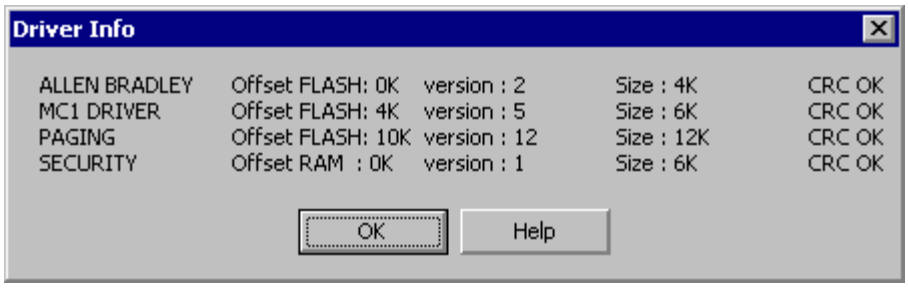
View - PC Comms Statistics

Displays communication successes and fails for all messages sent by Toolbox. Select the **Reset** button to clear totals.



View - Read Drivers Info

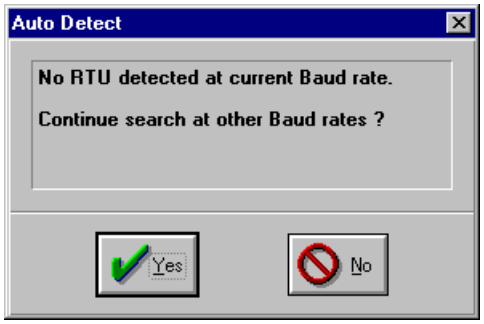
Displays information about the firmware drivers loaded in the RTU.


Driver Info
ALLEN BRADLEY Offset FLASH: 0K version : 2 Size : 4K CRC OK
MC1 DRIVER Offset FLASH: 4K version : 5 Size : 6K CRC OK
PAGING Offset FLASH: 10K version : 12 Size : 12K CRC OK
SECURITY Offset RAM : 0K version : 1 Size : 6K CRC OK
OK
Help

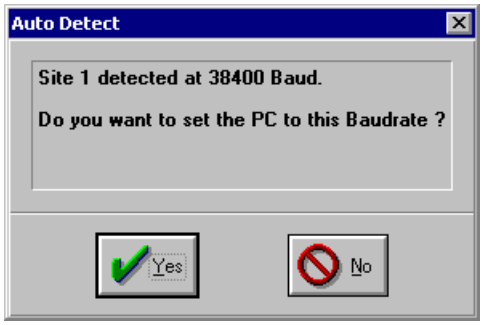
If a driver has been partly overwritten or has an error, “CRC Fail” will be displayed and a “Driver Error” system flag will be displayed in the RTU Status.

View - Auto Detect

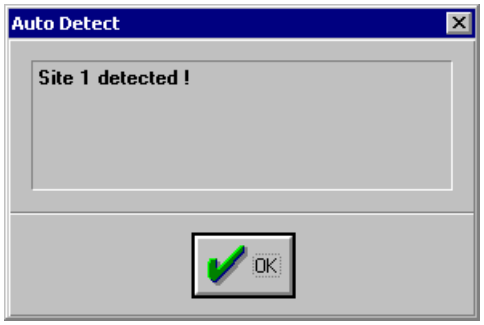
Detects the address and baud rate of the local RTU. Auto Detect will first try to communicate with the local RTU at the baud rate configured in Configuration, PC Setup.



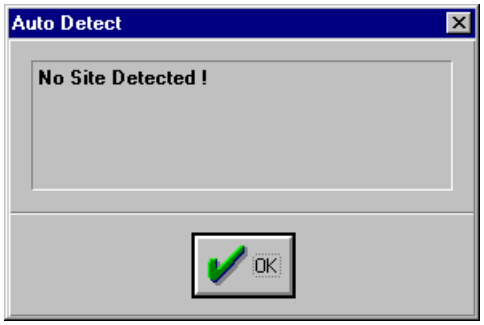
Baud rate of Toolbox does not match the local RTU.
Select **Yes** to check other baud rates.



Address and baud rate of the local RTU has been detected.
Select **Yes** to update Toolbox.



Baud rate of Toolbox matches local RTU.
Auto Detect returns address of the local RTU.



Toolbox could not detect the local RTU at any baud rate. Reasons:

- Incorrect COM port or IP Address (if using Ethernet) configured in Configuration, PC Setup
- The PC cable is not connected or is faulty.
- The PC cable is not connected to an RTU serial (or Ethernet) port.
- The RTU port has been incorrectly configured. The RTU configuration and ladder logic can be cleared by removing the battery link on the back of the CPU module and waiting for about 2 minutes. Note: the battery link for an LP-1/2/3 is inside the case.

9. Utilities Menu

Unlock RTU Port	Used to access a secured RTU port
Diagnostic Test	Checks RTU is operating correctly
Set Real-Time Clock	Allows the RTU's time and date to be set
Warm Start RTU	Equivalent to power resetting the RTU
Comms Analyser	View messages received and transmitted on a specified port
Comms Terminal	View messages received on any port and send AT commands to relevant devices
Dial Site, Hangup Site	Manually dial an RTU in the network list using a PSTN modem
Carrier Test	Used to test radio or private line ports
Upload RTU Variables to file, Download RTU Variables from file	Save local register values into a text file on a PC
Enable / Disable IO Scanning	Suspend updating of inputs and outputs
Enable / Disable Logic Processing	Suspend processing of ladder logic
Advanced	Advanced features eg. download firmware and drivers, cold start

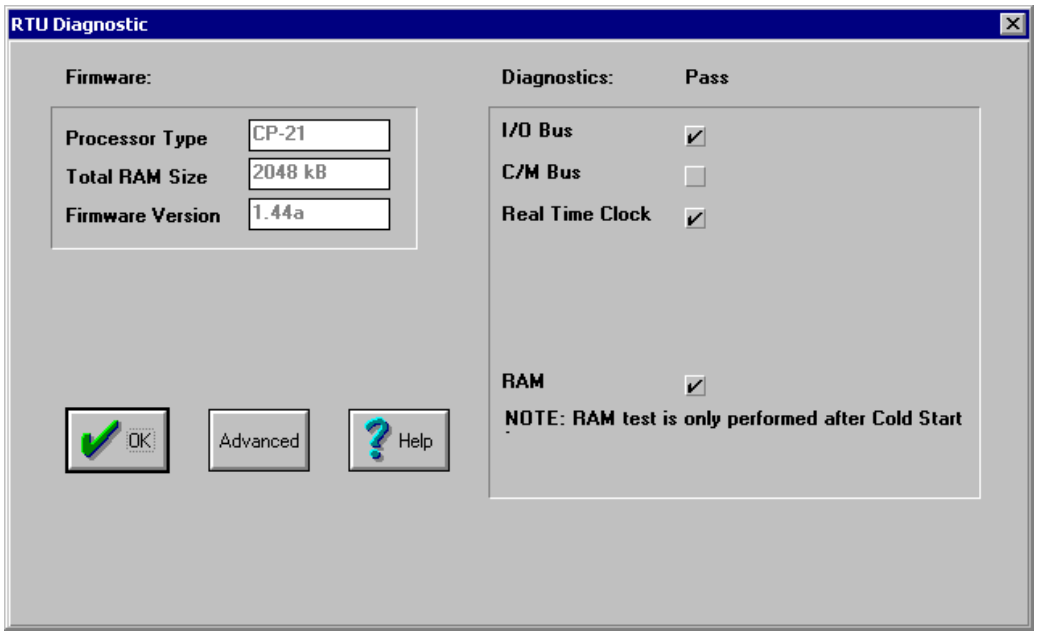
Utilities - Unlock RTU Port

Changes the security level of the RTU port to equal the security level of Toolbox (provided the Toolbox security level offers greater access). Changing the security level to 0 will allow an RTU to be re-configured.

After two minutes of comms inactivity, the RTU will automatically switch the port back to its configured security level.

Please see the appendix RTU Security for more information.

Utilities - Diagnostic Test



A tick indicates that the diagnostic test has passed (is successful). The test for the RAM Chips is only carried out after a cold start. The other diagnostic tests are continually updated as the RTU runs.

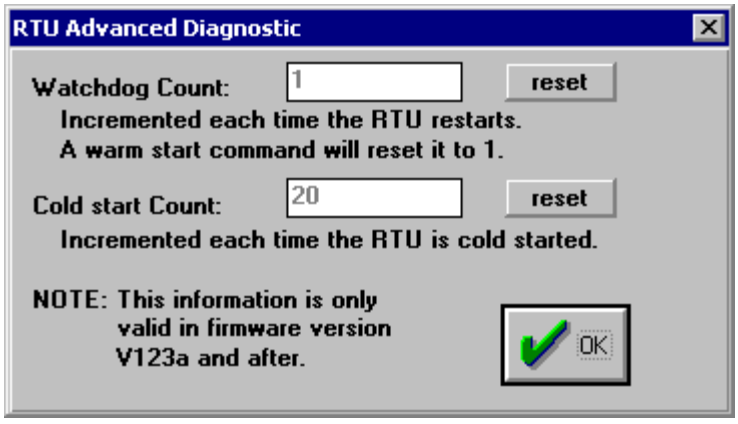
I/O Bus: Test is passed if an IO module is detected. If the CPU loses communications with an IO module or is unable to detect an IO module, the IO bus is denoted as unknown (blank).

C/M Bus: (Communications Bus) Test is passed if a communications module is detected (such as an MC module). If the CPU loses communications with a communications module or is unable to detect a communications module, the C/M bus is denoted as unknown (blank).

Real Time Clock: Test is passed if the seconds field of the real time clock has changed within 2 seconds. Otherwise displayed as failed.

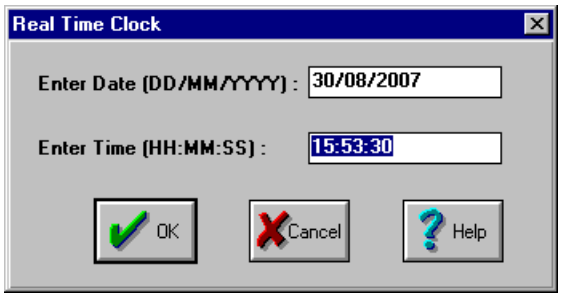
RAM: Passed if the battery backed RAM is OK.

Selecting the **Advanced** button on the Diagnostic window displays the following information:


RTU Advanced Diagnostic
Watchdog Count:
1
reset
Incremented each time the RTU restarts.
A warm start command will reset it to 1.
Cold start Count:
20
reset
Incremented each time the RTU is cold started.
NOTE: This information is only valid in firmware version V123a and after.
OK

The Advanced Diagnostic window displays the number of restarts (Watchdog Count) of the RTU and the number of cold starts. The Watchdog count is incremented each time the RTU is powered up or the Watchdog timer forces a restart or #YSYS.WARMST is set using ladder logic. Both the Watchdog Count and the Cold Start Count can be reset to 0 by selecting the respective **Reset** button.

Utilities - Set Real-Time Clock

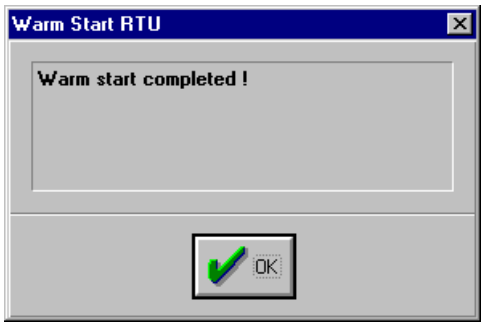


When **Set Real-Time Clock** is selected, Toolbox reads the PC's time and date and these values are used for the default settings. The defaults can be changed and then downloaded to the RTU by selecting **OK**.

Correct RTU time and date can be important when using event logs or other ladder functions. Some processors (eg. LP-1/2/3) will behave erratically unless the clock is set to a valid setting.

The clock can also be set from ladder logic using clock registers.

Utilities - Warm Start RTU



A Warm Start is equivalent to a power reset which causes the RTU to perform self-diagnostics and to set communication buffers to their default states.

A warm start retains the IO module inputs and outputs, local and network register values, event logs, RTU configuration and ladder logic.

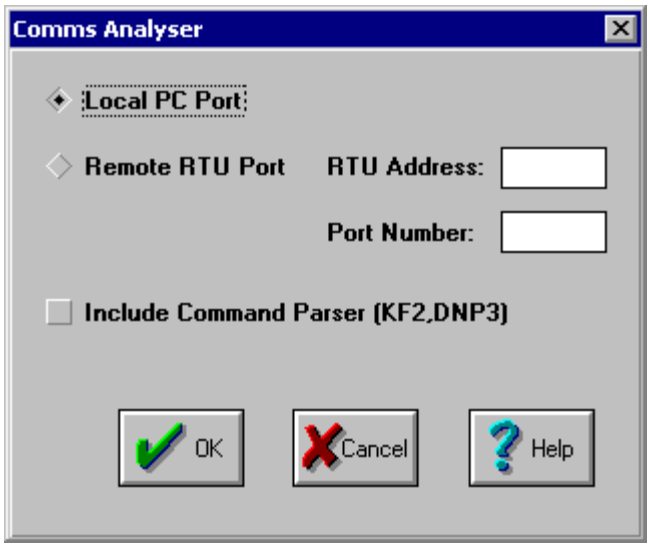
A warm start will automatically occur after downloading the RTU configuration but not after downloading ladder logic.

A warm start can also be triggered from ladder logic using the system register #YSYS.WARMST.

Utilities - Comms Analyser

The Comms Analyser displays all the messages received and transmitted on a specified port in hexadecimal format. The analysed port MUST BE DIFFERENT to the port used to send analyser messages to Toolbox.

Caution! Communications will be overloaded if analysing the radio port of a remote RTU if the same radio port is used to send the analyser messages back to Toolbox. Communications can be restored by performing a carrier test on the local radio port and then sending a warm start to the remote RTU while the carrier test is in progress.



Comms Analyser can be used for the *Local PC Port* (which uses the COM port setting in Configuration, PC Setup), or for a *Remote RTU Port*. To access a remote RTU port, a valid RTU address (1-249) and port number (1-16) must be entered. *Include KF2 Command Parser* can also be selected which will interpret the characters into Kingfisher Series 2 messages. This will display the target and initiating RTU addresses, the Kingfisher Series 2 command number and a message description.

Comms Analyser will display the last 16 messages (approx.) with the newest message displayed at the top of the window. All messages are also written to the file - ANALYSER.TXT in the program file folder (eg. C:\Program Files\Kingfisher\Kingfisher Toolbox\ANALYSER.TXT). The analyser display can be paused by pressing CTRL-Q. When CTRL-Q is pressed again, the analyser display will continue updating.

When monitoring a CPU port, the analyser will detect any *character* being received or transmitted (even if it is part of an unrecognised protocol). When monitoring an MC port, the analyser will only detect *complete protocol messages* of the protocol which has been configured for that port.

Care should be taken when using the Remote Comms Analyser in a busy network, as it will generate a significant amount of network traffic (slightly more than the amount of traffic on the port being monitored).

The comms analyser works best when all other Toolbox communications windows are closed (eg. RTU Status).

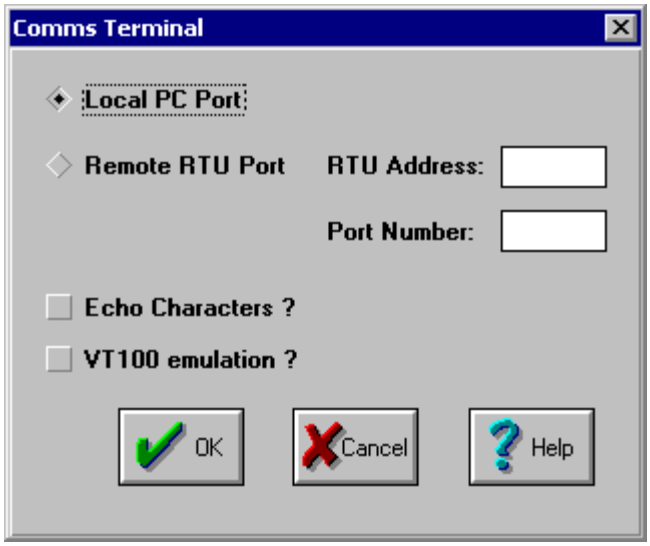
Caution! The analyser window or Toolbox must be closed *before disconnecting* from the RTU otherwise the RTU will continue to broadcast the messages from the monitored port indefinitely. Closing the analyser screen causes a *stop analyser mode* message to be sent to the monitored RTU. Analyser mode can be forced to stop by restarting the Toolbox analyser for the relevant RTU port and then closing the analyser again.

Utilities - Comms Terminal

Comms Terminal can be used view characters that are *received* by the local PC COM port or by any RTU port in the telemetry system. Comms Terminal can also be used to *send* AT commands to devices that support the AT command set.

Comms Terminal can be used with a local PSTN modem or GSM (connected to the PC) to dial into a remote RTU for diagnostics and configuration.

Ethernet messages between RTUs can be viewed using the *Wireshark* program and a PC that is connected to the same Ethernet network. To download a free copy of Wireshark, visit www.wireshark.org. A Kingfisher version of Wireshark is also available from http://helpdesk.servelec-semaphore.com that has support for deciphering the Kingfisher protocol.



Comms Terminal can be used for the **Local PC Port** (which uses the COM port setting in Configuration, PC Setup), or for a **Remote RTU Port**. To access a remote RTU port, a valid **RTU Address** (1-249) and **Port Number** (1-16) must be entered.

Echo Characters can also be selected, which will echo any send characters back to the display window.

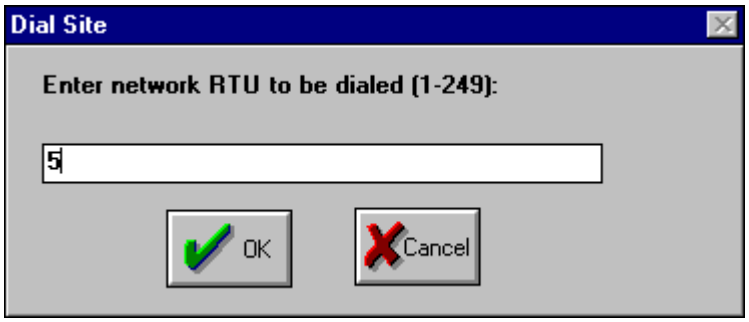
Once the port has been specified, a terminal window is then opened. ASCII characters can be sent to the specified port (CPU or MC ports) and characters received can be viewed (for CPU only). Characters received on MC ports cannot be viewed.

Comms Terminal is very useful for communicating with dial option boards or local PSTN modems. Some Hayes AT modem commands and responses are shown below.

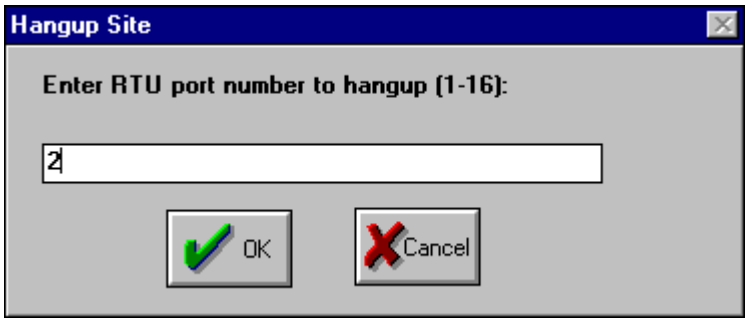
Command	Description
AT	General command to check if modem is OK. Modem should respond with **OK** (if verbose on) or **0** (if verbose off).
ATI	Modem should respond with model information
ATDT 03 9123 4567	Number to be tone dialed (phone number of remote RTU)
ATDL	Dial last number again
CARRIER 9600 PROTOCOL: LAP-M COMPRESSION: V.42BIS CONNECT 9600	Connection information. After the CONNECT 9600 statement appears, Toolbox can be used to view the RTU status, download a configuration or perform any other function that would be possible if Toolbox were directly connected to the remote RTU with a cable. If Toolbox is used to interrogate the RTU before this message appears, communication problems may occur.
+++	Tells the modem that the next characters are a command. Modem will respond with **OK** (or **0**) when ready for an AT command.
ATH	Hang up modem. Modem will respond with **OK** (or **0**) when disconnected.

For diagnosing faults when using a PSTN modem or GSM, please see the appendix - RTU Commissioning, Trouble Shooting.

Utilities - Dial Site / Hangup Site

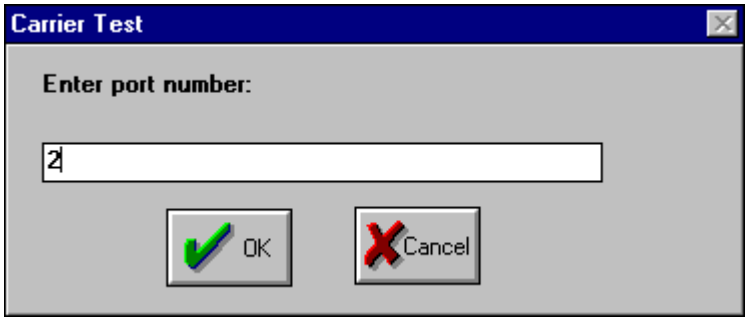


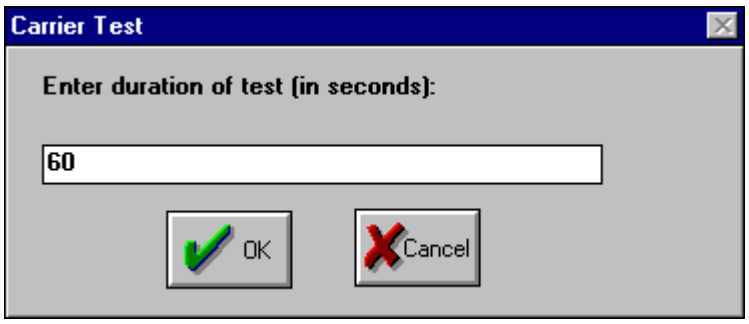
Dial Site: tells the local RTU to dial a remote RTU (1-249). The local RTU will dial the phone number configured for that remote RTU in **Configuration, Phone Directory**. The port that is used for dialing is defined by the network link for that remote RTU.



Hangup Site: tells the local RTU hangup the PSTN modem on port 1-16.

Utilities - Carrier Test





A carrier test will force the specified radio or private line port (1-16) to transmit for a number of seconds (1-999). A carrier test will not operate on serial ports. A carrier test is very useful when wanting to test the radio signal strength received from a remote RTU. By forcing the remote RTU to transmit continuously for say 60 seconds, the RX power at the local RTU can be measured. This also identifies which RTU is transmitting since the other remote RTUs will be locked out during the transmission.

Caution! Radio carrier tests should not be longer than 60 seconds in order to prevent overheating and damage to the radio.

Utilities - Upload/Download RTU Variables

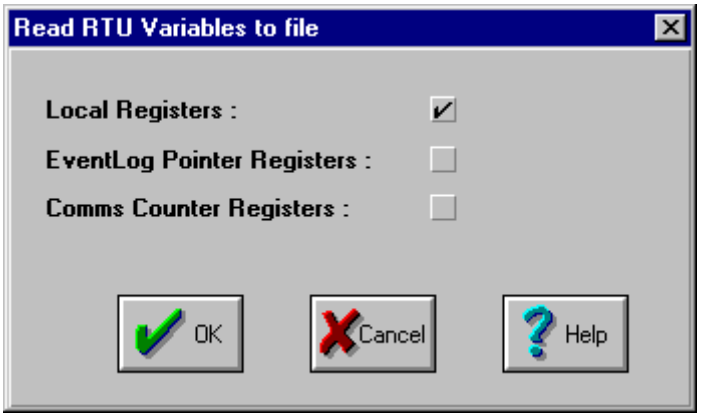
Local register values can be uploaded (or saved) into a text file on a PC and then downloaded back into the RTU. This is useful when an RTU needs to be cold-started or have new firmware loaded as the contents of all the local registers are cleared.

Local registers (#R1 to #R2048), event log pointer registers (#YLLOGIDX1 to 255) and communication fail (#YLFAIL) and success (#YLSUCC) registers can all be read from the RTU and saved to a file.

The register file is stored in tab-delimited ASCII format and can be edited using any text editor. When editing the file, registers may be removed, added or have their values changed to any positive integer in the range of 0 to 65535. Only registers remaining in the file will be downloaded into the RTU. Comments can also be added to the file by beginning the comment with a semicolon (;). Comments can be inserted on new lines or added to the end of existing lines.

Example:

- Select **Upload RTU Variables to file** and then choose a filename.
- If the file does not exist, Toolbox displays the window below. If the file does exist, then only the registers specified in the file will be uploaded.



- After completing the cold-start or the firmware upgrade, select **Download RTU Variables from file**, then choose the same file to download the registers back into the RTU.

Note: this function enables the user to overwrite local register values. If the logic includes things like accumulated totals, edge triggers or register mapping, these may be overwritten or triggered when uploading and downloading RTU variables.

Utilities - Enable / Disable I/O Scanning

An RTU will automatically attempt to scan all its IO modules at the configured IO Scan Interval (**Configuration, System Parameters**). The RTU can freeze its inputs and outputs by disabling IO scanning.

Utilities - Enable / Disable Logic Processing

An RTU will attempt to processes its ladder logic at the configured IO Scan Interval (**Configuration, System Parameters**). Processing of Ladder logic can be stopped by selecting **Disable Logic Processing**. The status of logic processing can be viewed in the RTU Status. Note: the RTU Status will also display “Logic Processing: Disabled” if the RTU does not have any ladder logic loaded or if ladder logic has been cleared from memory (eg. after a cold start).

Utilities - Advanced

Options: Download CPU Firmware, Download MC Firmware, Download Firmware Driver, Upload Configuration, Cold Start, Swap Master, PC System ID, Read/Write System Register and Upload Memory.

Utilities - Download CPU Firmware

Firmware in CPU modules can be upgraded to add new features and enhancements.

The firmware version that is running in the CPU module is displayed in the RTU Status window.

Caution! An RTU is cold started after downloading CPU firmware. Firmware can be downloaded to a *remote* RTU but it is possible that communications will be lost after the RTU is cold started. Note: the RTU will remember the configuration settings for its ports as follows;

- PC-1 - Ports 1-8
- CP-11/21 - Ports 1-4
- CP-12 - Ports 1-4

Do not hot swap any module on a connected rack while downloading firmware.

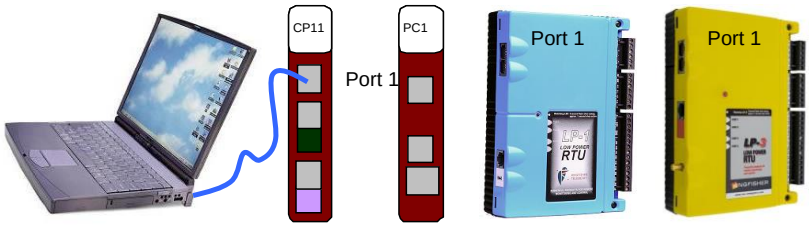
MC Firmware must be downloaded via the first port of any MC module.

Downloading processor firmware or drivers requires a reliable connection to a port that is retained after a cold start.

Configuration, logic and ladder can be downloaded via any port, including via remote RTUs, providing a reliable connection exists.

1. To download firmware to the *local* CPU, connect to Port 1 of the CPU.

To download firmware to a *remote* CPU, connect to any serial port on the local RTU (a communications link is required between the local RTU and the remote CPU).



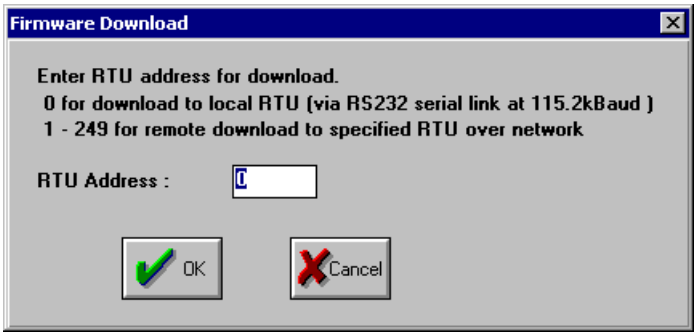
Redundant Processors: please ensure only one processor is installed on the backplane while upgrading firmware

2. Select **Utilities, Advanced, Download CPU Firmware**

3. To download firmware to the *local* CPU set **RTU address** to 0 and then select **OK**

To download firmware to a *remote* CPU set **RTU address** to the address of the remote CPU and then select **OK**

Toolbox will check if it can communicate with the RTU address specified.

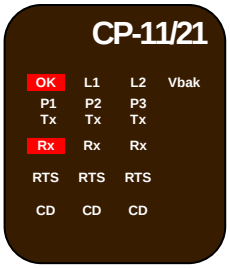

Firmware Download
Enter RTU address for download.
0 for download to local RTU (via RS232 serial link at 115.2kBaud)
1 - 249 for remote download to specified RTU over network
RTU Address :
0
OK
Cancel

4. Select the CPU firmware file to download and then select **OK**

Note: LP-1 firmware is different to LP-2/3 firmware.

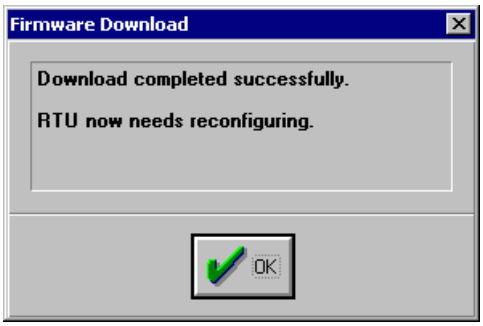
MODULE	FIRMWARE NAME	EXAMPLE
PC-1	Vxxxx.HEX	V143D.HEX
CP-10/11	Vxxxx.H32	V144E.H32
CP-21	Vxxxx.H21	V146B.H21
LP-1	LP1Fxxxx.BIN	LP1F143A.BIN
LP-2/3	LP23Fxxxx.BIN	LP23F150A.BIN
CP-12	CP-12 xxxx.cp12	CP-12 1.47c.cp12

5. While downloading firmware *locally*, the **Rx** LED for Port 1 of a PC-1 or CP-10/11/12/21 will stay on.



6. After a successful download the following message will be displayed.

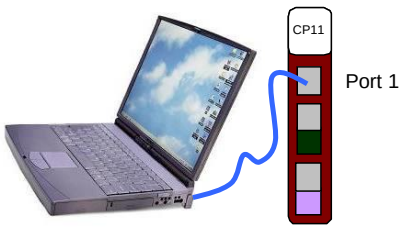
Note: The RTU configuration will now need to be downloaded since downloading firmware cold starts the RTU and clears the existing configuration.



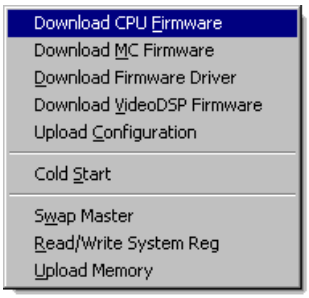
Recovering A CP-10/11 After Downloading MC Firmware

An MC-10/11/12 uses the same circuit board as a CP-10/11 and so it is possible to download MC firmware into a CP module. This will change a CP-10/11 into an MC-10/11/12 (even though the module lens still says **CP-10/11**!) To recover a CP-10/11 after downloading MC firmware, please follow the steps below.

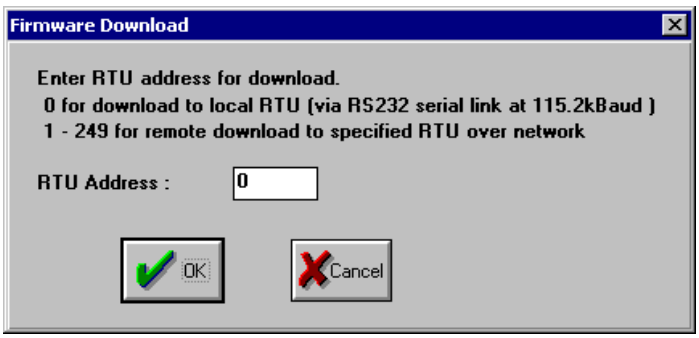
1. Connect to Port 1 of the CP-10/11 module (that contains the MC firmware)



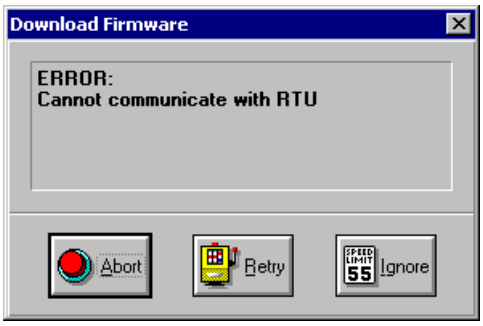
2. Select **Utilities, Advanced, Download CPU Firmware**



3. Set *RTU address* to 0 and then select **OK**

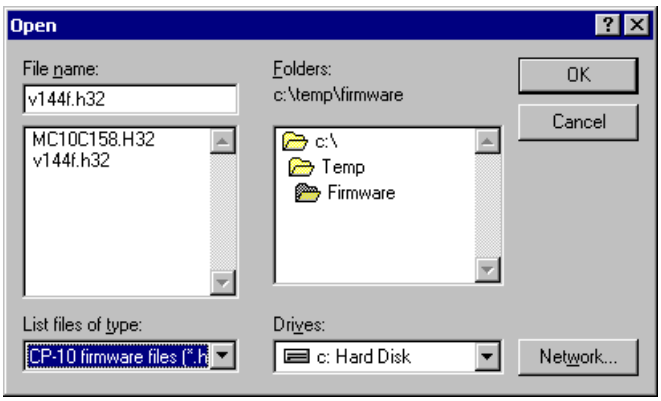

Firmware Download
Enter RTU address for download.
0 for download to local RTU (via RS232 serial link at 115.2kBaud)
1 - 249 for remote download to specified RTU over network
RTU Address :
0
OK
Cancel

4. When the message "ERROR: Cannot Communicate with RTU" appears select **Ignore**



5. Select the *CP-10/11* firmware file to download and then select **OK**

Note: ensure **List files of type** is set to **CP-10 firmware files** (as shown highlighted)



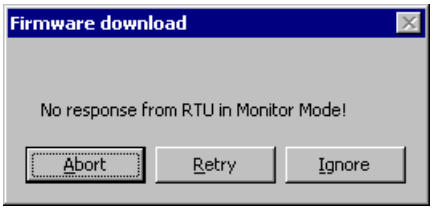
6. When the message "Download speed will be set to 115.2k Baud!" appears select **OK**

Firmware Download
Download C:\Temp\Firmware\v144f.h32 now ?
Note: Firmware will be sent to the local RTU only. Connection must use port #1.
Download speed will be set to 115.2kBaud!
OK
Cancel

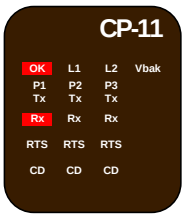
7. If the message "No response from RTU in Monitor Mode!" appears, select **Abort**

Toolbox 32 is using the wrong baud rate. From **Configuration, PC Setup** select another Baud Rate and return to step 2 above.

Note: the baud rate of the RTU port can be determined from the configuration file that was last running in the CP-10/11.

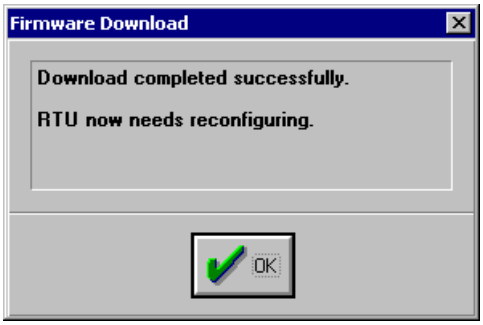


8. While downloading firmware *locally*, the **Rx** LED for Port 1 of the CP-10/11 will stay on.



9. After a successful download the following message will be displayed.

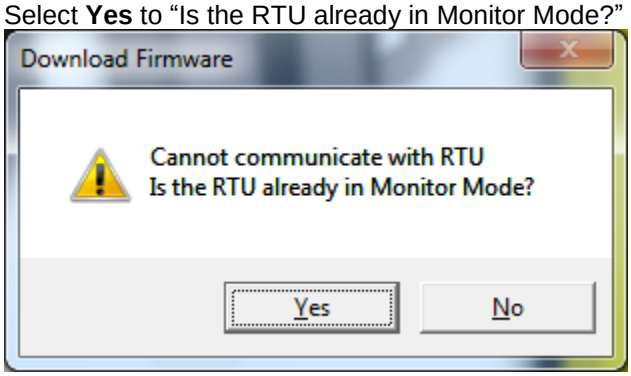
Note: The RTU configuration will now need to be downloaded since downloading firmware cold starts the RTU and clears the existing configuration.



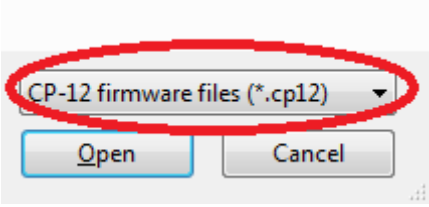
Recovering a CP-12 after downloading MC-12 firmware

Similar to the MC-10/11/12 the MC-12 shares the same circuitry as a CP-12, therefore it's possible to convert a MC-12 to and from a CP-12. The steps below outline how to do so.

1. Place the module into monitor mode, refer to Appendix – Clearing configuration with Monitor Mode
2. Select **Utilities->Advanced->Download CPU Firmware**
3. Set *RTU address* to 0 and then select **OK**
4. Select **Yes** to "Is the RTU already in Monitor Mode?"



5. Chang the filter to "CP-12 firmware files" in the Open Dialog



6. When the message "Download speed will be set to 115.2k Baud!" appears select **OK**
7. When the download has completed hit finish.

Note: The RTU configuration will now need to be downloaded since downloading firmware cold starts the RTU and clears the existing configuration.

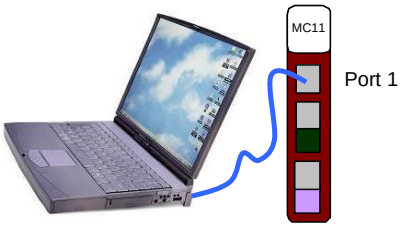
Utilities - Download MC Firmware

Firmware in MC modules can be upgraded to add new features and enhancements.

The firmware version that is running in an MC module can be determined from the hardware overview (View, Hardware Overview) by selecting the button next to the MC module.

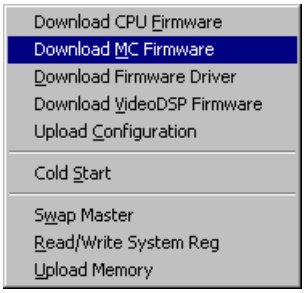
1. To download firmware to the *local* MC module, connect to Port 1 of the **MC** module.

(Note: firmware cannot be downloaded to a remote MC module)



⚠ **Caution!** Do not connect to the *CP-10/11*. Toolbox 32 allows MC firmware to be downloaded into a CP-10/11. If this has occurred, please see the previous topic - Recovering A CP-10/11 After Downloading MC Firmware.

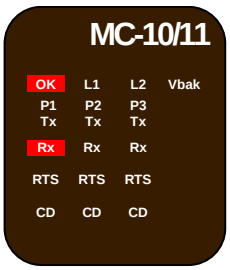
2. Select **Utilities, Advanced, Download MC Firmware**



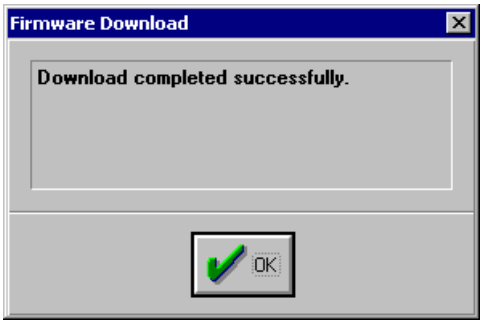
3. Select the MC firmware file to download and then select **OK**

MODULE	FIRMWARE NAME	EXAMPLE
MC-1	MC1_Cxx.HEX	MC1_C30.HEX
MC-10/11	MC10Cxxx.H32	MC10C158.H32

4. While downloading firmware, the **Rx** LED for Port 1 of the MC module will stay on.



5. After a successful download the following message will be displayed.



Utilities - Download Firmware Driver

Drivers can be downloaded into flash memory (non-volatile) or SRAM (battery backed). Drivers in SRAM are cleared after a cold-start while drivers in flash memory are preserved after a cold-start. Drivers can be downloaded locally (by plugging into the RTU) or remotely (over the RTU communications network).

Caution! An RTU is cold started after downloading a driver to *flash* memory (the RTU does not cold start after downloading to SRAM). The RTU configuration and logic must be downloaded after a cold start. Drivers can be downloaded to flash memory in a *remote* RTU but it is possible that communications will be lost after the RTU is cold started. Note: the RTU will remember the configuration settings for the first four CP-11/12/21 ports or first eight PC-1/CP-1 ports.

- Downloading new *firmware* clears all drivers in flash memory and SRAM and therefore new firmware should be downloaded *before* downloading any drivers. Please ensure the RTU is running the minimum required firmware by checking the driver listing in **protocols.pdf** available from http://helpdesk.servelec-semaphore.com.
- It is recommended that drivers are downloaded to flash memory if space is available. Available driver memory:

CPU Type	Flash Memory (kB)	SRAM (kB)
PC-1	28	Configurable
CP-1	32	Configurable
CP-10/11/12	64	Configurable
CP-21*	N/A	N/A
LP-1/2/3	0	Configurable

  * All CP-21 drivers are included in firmware and are not downloaded separately.
- Before downloading drivers into SRAM (if required), memory space must be allocated in the RTU configuration (please see the topic Configuration - Memory, Firmware Drivers) and the RTU configuration downloaded into the RTU. If drivers are also to be downloaded to flash memory, these should be downloaded *before* downloading the RTU configuration (since the RTU is cold-started after downloading each driver to flash memory).
- After selecting **Utilities, Advanced, Download Firmware Driver**, Toolbox will attempt to communicate with the RTU to determine its CPU type. If communications are successful, Toolbox will allow the firmware driver to be selected for downloading (PC-1/CP-1 drivers use the file extension DRV, CP-10/11 drivers use the file extension D32 and LP-1/2/3 drivers use the file extension DHI).
- The first driver is downloaded (into flash memory or SRAM) using an address offset of 0 kB. The second driver is downloaded after the first driver by using an address offset greater than or equal to the total size of the previous drivers (memory space can be left between drivers). An example of downloading 3 drivers into a CP-10/11 is detailed below. Note: when downloading drivers into a PC-1 / CP-1, the maximum address offset that can be used is 15K and so the largest driver can be downloaded last.

Driver Name	File Size (kB)	Offset (kB)
TXUPDATE.D32	6	0
PAGING11.D32	12	6
MODBUS03.D32	10	18

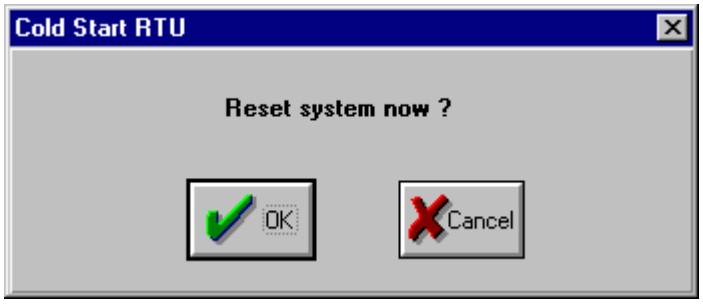
Note 1: The DRIVERS.VER file is used by Toolbox to ensure the RTU has the minimum required firmware before allowing a driver to be downloaded. Please ensure that the latest DRIVERS.VER file is located in the Toolbox program files folder before downloading firmware drivers. The latest versions of DRIVERS.VER and standard firmware drivers (eg. Paging, TX Update, RX Update and Modbus) are all available from the Semaphore web site http://helpdesk.servelec-semaphore.com.
Note 2: The CP-1 redundancy driver is downloaded into a reserved area of flash memory and does not use any of the flash memory allocated for standard drivers.
Note 3: Each CP-1 redundancy driver is designed to work with the corresponding firmware version. Eg. The redundancy driver *red_139e.drv* is used with firmware version 1.39e.

Utilities - Upload Configuration

This option will upload all the SDB configuration file settings from the RTU (except for the site description and TMR directory as these are not stored in the RTU). Ladder Logic can also be uploaded from the RTU if the ladder edit file (FILENAME.LL) was stored in the RTU. The uploaded SDB information can be saved as a configuration file by selecting **File, Save As** and entering a filename.

Note: ladder logic is stored in the RTU by checking the Store ladder logic files in RTU box in **Configuration, Memory** and then downloading the RTU Configuration. Ladder logic can then be uploaded from the RTU by using **Logic, Advanced, Upload .LL File from RTU**. The **Logic** menu is available when an SDB file is open (displayed in front).

Utilities - Cold Start



A cold start clears all the RAM (including the registers, event logs and ladder logic) and sets all the IO modules to their default state. A cold start is recommended before downloading a new configuration into a local RTU.

Caution! Cold starting a remote RTU is not recommended. After a cold start, an RTU will remember the configuration settings for the first 4 ports (8 ports for a PC-1/CP-1) but it is possible that communications will be lost after the cold start.

Utilities - Swap Master

When using redundant CPUs, this command will swap the duty CPU and the standby CPU. For more information, please see the appendix - Redundancy.

Utilities - PC System ID

This option is no longer available in Toolbox 1.44d or newer! When communicating with RTUs using Toolbox, the PC is treated like another RTU. An RTU will only respond to messages with a system ID of AE (default setting) or to messages using the RTU's System ID.

Utilities - Read/Write System Reg

Advanced use.

Utilities - Upload Memory

Advanced use. Reads data from an RTU memory location and writes it to two files <filename>.BIN and <filename>.TXT. The two files are stored in the Toolbox program folder. <filename> should be specified without a path or extension. Eg. **MEMORY1** not **C:\MEMORY1.TXT**

10. Appendices

Printing Ladder Logic	How to print ladder logic
RTU Security	How to secure an RTU against unauthorised Toolbox users
Hexadecimal Numbers	Explanation of what hexadecimal numbers are
Redundancy	How to setup an RTU to use redundant CPUs, power supplies and communications
Version Control	Only relevant for very old versions of Toolbox configurations
IEC Compliant Register Names	IEC compliant names for local registers and other data
Series I RTUs	How to use current Kingfisher RTUs with Series I RTUs
Calibrating RTU Modules	Setting up special module options ie. IO-4 strain gauge and RT-1 inputs
RTU Commissioning	Items to check when setting up an RTU for operation in the field
Ethernet	Explanation of how Ethernet is implemented including settings and parameters
Communicating With A G3 RTU	How to communicate with a G3 RTU
RTU Data	How to address RTU registers and system information using ladder logic.
Potential DEP Issue	Potential DEP Issues on Windows Vista, Windows 7 or Windows Server 2008.

Appendix - Printing Ladder Logic

Printing Ladder Logic Using A Text Editor

Ladder logic can be printed to a text-only file by first displaying ladder logic in Toolbox using text characters (the display mode can be toggled between line draw characters and text characters using the menu **File, Select LineDraw/Text Chars**). Once the ladder is displayed as text characters, select **File, Print To File**. FILENAME.PRN will then be a file containing standard text characters and can be opened and printed using a basic text editor (eg.Microsoft Write, Wordpad).

Printing Ladder Logic Using Microsoft Office

A macro has been created for Microsoft Word that automatically converts and formats a ladder PRN file ready for printing. The document containing the macro is available from the Semaphore website http://helpdesk.servelec-semaphore.com.

- Using Toolbox, open the ladder logic configuration to print. Select **File, Print To Text File**. [Filename]**.PRN** will then be created.
- Check that macros are enabled in Word by selecting **Tools, Macro, Security**. **Medium** or **Low** should be selected.
- Open the document *PrintingLadderLogic.DOC* using Word.
- Insert the [Filename].PRN file into the document by selecting **Insert, File…**
  Word 2000+: When asked **Convert file from** select **Text Only**.
  Word 97: When asked **Select the Encoding that makes your document readable** select **Windows (default)**.
- Run the ladder conversion macro included with the document by selecting **Tools, Macro, Macros**. Double-select **ConvertLadderSymbols**.
- Ladder is now ready to print (**File, Print**).

Appendix - RTU Security

An RTU can be secured against unauthorised re-configuration or unauthorised access to data registers.

An RTU is secured by loading the security driver (as detailed in the next topic) and configuring each communications port with a non-zero security level using Toolbox (the 5 levels of security are detailed below).

The security driver limits access to RTU registers and configuration parameters by limiting messages received from Toolbox or from another RTU or device using the Kingfisher or Modbus protocols. Note: other protocols are not affected by the security system and can read and write RTU data. However, only the Kingfisher protocol can be used to reconfigure the RTU if permitted by the security level of the communications port or if the communications port is first unlocked using Toolbox.

The security levels that can be configured for each port are as follows:

RTU Port Security Level	Read Access	Write Access	Notes
0	Everything	Everything	Highest access level. Only this level can be used to download a configuration and reconfigure the RTU.
1	Everything	Everything except System Registers and Ladder Logic	System Registers include all configuration parameters.
2	Everything	#R1 to #R512 only	
3	Everything	Nothing	
4	#R1 to #R512 only	Nothing	
5	Nothing	Nothing	RTU can only be accessed by first unlocking the port using Toolbox. Note: Ethernet port will still respond to a ping request.

Each RTU Communication Port has a default security level of 0 (full access). Other security levels may be configured from **Configuration, Port List**. By configuring all RTU ports in a network with security levels of 3, 4 or 5, the network is then secure against unauthorised reconfiguration and can only be reconfigured by an authorised Toolbox user after unlocking the RTU comms port as detailed in the next section. If the security driver is not loaded, all ports default to full access (level 0).

Toolbox Security Levels

By default, a username and password is not required when running Toolbox. However, to access a secured RTU, the Toolbox security login must be enabled and then a configured Username and Password entered as illustrated.

To enable the Toolbox security login, run the **Toolbox Security** program (supplied with Toolbox). When asked "Enter Access Code" enter *actionuser*. Select the plus (**+**) button and add the username *admin* and a password (eg. *admin*). The admin password will then become the new Kingfisher Security Access Code. Once the **admin** username is added, the security system will be enabled. When Toolbox is next run, a Username and Password will then be requested.

The **Toolbox Security** program allows new usernames and passwords to be created each with their own security level. The default Usernames and Passwords supplied with Toolbox Security are shown below.

Username	Password	Security Level	Toolbox Functionality and Menu Options
action0	action0	0	Full access
action1	action1	1	Full access except downloading an RTU configuration
action2	Action2	2	Can view configuration but cannot make any changes
action3	Action3	3	Same as level 2
action4	Action4	4	Cannot view or change the RTU configuration

Configuring RTU Security

- First ensure that the Kingfisher Security driver (SECURITYxx.Dxx) is supported by the type of RTU that is being used and that the latest firmware is loaded in the RTU (as detailed in **protocols.pdf** available from http://helpdesk.servelec-semaphore.com).
- Whenever passwords, usernames or security levels are changed (using the Toolbox Security program), it is necessary to re-download the security driver into each RTU. Note: the security driver must not be read-only otherwise the security list cannot be changed. The security driver properties can be viewed and changed by right-clicking the driver using Windows Explorer, selecting *Properties* and then unchecking the read-only box. A custom Username (up to 16 characters), Password (case-sensitive, up to 16 characters) and Security Level (0-5) can be defined for each user by running the Kingfisher Security program.
- Download the security driver into the RTU. Note: the passwords configured using the Kingfisher Security program are embedded into the security driver when it is downloaded. The RTU uses these embedded passwords to ensure that security messages are valid. For a CP-21, the security driver will need to be downloaded into SRAM *after* downloading the RTU configuration in the next step below.
- Setup port security by configuring the Port List in the RTU configuration using Toolbox. Set *Port Security* for each port from **Configuration, Port List**. Once port security has been configured, download the configuration into the RTU. Note: if using a CP-21, ensure **Configuration, Memory** *Firmware Drivers* is set to 8 K.
- A secured RTU port can then be accessed by using the menu **Utilities, Unlock RTU Port**. This command changes the security level of the RTU port to equal the security level of the Toolbox security login (provided the Toolbox security level offers greater access). After two minutes of comms inactivity, the RTU will automatically switch the port back to the configured security level.

Recovering A Secured RTU

For the highest level of security, the security driver can be loaded in flash memory (as usual). However, if the passwords are lost and the port security level is 2 or higher, it will not be possible to reconfigure the RTU and it may not be possible to communicate with the RTU (if the security level is 5). The RTU can be recovered if passwords are lost by downloading firmware locally into port 1 of the CPU. Alternatively, the security driver can be loaded in SRAM (the recommended method). There are two reasons for this:

- If the user database is updated and the security driver is re-downloaded into SRAM, the RTU will keep its configuration. When the driver is downloaded into flash memory, the RTU is cold started and loses its configuration.
- If passwords are lost, it is possible to recover the system by removing the CPU battery link and clearing the SRAM. This will erase the driver and allow full access to the RTU. If the driver is loaded into flash memory, it will not be erased by removing the battery link and the RTU will remain password protected.

Security Audit Trail

Whenever an RTU receives a command to unlock one of its ports, an event log is generated (if the RTU has memory configured for event logs) called *System Log*. The *usertype* of the event log is set to the index number of the user who unlocked the port. The index number is the number, or order, of the user in the user database (as displayed by the Kingfisher Security program or the *users.dbf* file). A value of 0 indicates the first user in the list, 1 indicates the second user etc.

Appendix - Hexadecimal Numbers

Hexadecimal is a numbering scheme similar to decimal but instead of counting from 0 to 9, hexadecimal counts from 0 to 15. The hexadecimal, decimal and binary equivalents are shown below.

Hex	Decimal	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111

Hex	Decimal	Binary
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

In Kingfisher RTUs, the mask value is a 4 digit hexadecimal value corresponding to channels 1 to 16. The mask used to enable one channel (or bit) only is as follows:

Ch1	0001 Hex	Ch9	0100 Hex
Ch2	0002 Hex	Ch10	0200 Hex
Ch3	0004 Hex	Ch11	0400 Hex
Ch4	0008 Hex	Ch12	0800 Hex
Ch5	0010 Hex	Ch13	1000 Hex
Ch6	0020 Hex	Ch14	2000 Hex
Ch7	0040 Hex	Ch15	4000 Hex
Ch8	0080 Hex	Ch16	8000 Hex

To enable more than one channel at a time, the mask value must have each of the required channels set ON.

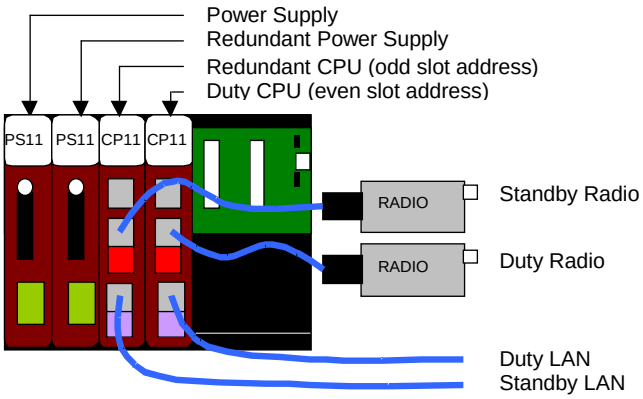
Examples:

To Enable Channels	Use Hex Number	Binary Equivalent
1, 2, 3	0007	0000 0000 0000 0111
1, 16	8001	1000 0000 0000 0001
1, 7, 8, 13	10C1	0001 0000 1100 0001

Hexadecimal numbers are specified in Kingfisher RTUs using the format 16#xxxx where xxxx is the hexadecimal number and can be 1 to 4 digits long.

Appendix - Redundancy

Redundancy allows an RTU to cope with equipment failures and continue operating normally. The RTU shown below has redundant power supplies, CPUs and communications. The setup of these components is detailed in the following sections. Another way of coping with equipment failures is to have two separate RTUs as detailed in the topic Redundant RTUs. Each RTU can also be monitored and controlled by two or more PCs running SCADA software as detailed in the topic Redundant PCs.



Redundant CPUs

An RTU can have two redundant CP-10/11/12/21 modules with one CPU in duty mode and one CPU in standby mode. If the duty CPU fails or is removed, the standby CPU takes over full control of the RTU.

A system with redundant CPUs must always have one CPU in an odd slot address and one CPU in an even slot address (note: the CPUs do not have to be physically next to each other on the backplane). When the system starts up, the CPU in the even slot will become the duty and the CPU in the odd slot will become the standby (indicated by the L2 LED flashing).

The duty CPU does all the same things as a single CPU system and also scans the standby CPU to check if it is still in standby mode and then updates the standby CPU's registers with its own values.

Updating The Standby CPU

- Digital and analog hardware registers are updated whenever a value is written to a digital or analog output module.
- Network data is updated whenever a new data message is received by the duty CPU.
- Local registers and event logs are updated every couple of seconds.
- The real time clock is synchronized every hour and whenever the duty clock is written to.
- Network data, timer registers, system registers, inactivity parameters, network configuration variables and most port configuration variables are checked a couple of times per minute and any differences are updated.
- Ladder logic, telephone numbers, Ethernet parameters (eg. IP address), PSTN parameters, paging parameters and TMR parameters *are not updated*.

Automatic Changeover

A standby CPU will turn into a duty CPU if:

- The standby CPU cannot hear any communications on the I/O bus and the communications bus for approximately 5 seconds.
- The duty CPU loses communications on the communications bus or the IO bus and forces a changeover.
- The value of the system register #YEXCEP is set to 16#800 from ladder logic. Note: a processor will not swap back to standby mode until it has been the duty processor for at least 60 seconds.
- A **Utilities, Advanced, Swap Master** command is issued from Toolbox to either the duty or to the standby CPU. Note: if the CPU that requests the swap gets no response from the other CPU, no action is taken as it is assumed there is no other CPU present.
- Ladder logic is disabled while downloading firmware drivers or after a cold start or firmware download. Note: the CPUs will not swap over if ladder is manually disabled using the Toolbox command **Utilities, Advanced, Enable/Disable, Disable Logic Processing**.
- A warm start command (**Utilities, Warm Start RTU**) is issued to either CPU when the standby CPU is in duty mode. This causes both CPUs to start up in their default state (the CPU in the even slot will become the duty).

Replacing CPUs

CPUs can be replaced after powering down the RTU or while the RTU is still running (not recommended). When a CPU is replaced, the RTU will automatically warm start itself.

- The standby CPU (installed in an odd slot address) can be replaced at any time without losing new data if it is running in standby mode (L2 LED is flashing).
- The duty CPU (installed in an even slot address) can be replaced at any time but any new data that is received by the standby CPU (now running in duty mode) will be over written. To avoid losing the new data, the replacement duty CPU can be forced to start up in standby mode, allowing the standby CPU to copy the new data to the replacement duty CPU. To force a CPU to start up in standby mode, set the system register 2471 Hex to 1 (using the menu **Utilities, Advanced, Read/Write System Reg**). This will cause a one-shot start up in standby mode and then the system register 2471 will be automatically cleared. After waiting about 5 minutes to ensure all the data has been updated in the replacement duty CPU, the CPUs can be swapped over (if desired) using the command **Utilities, Advanced, Swap Master**.

Configuring Redundant CPUs

- Redundant CPUs are configured *one* at a time by plugging one CPU into the backplane and then downloading firmware (optional), drivers (optional) and the RTU configuration. After removing the first CPU, the second CPU is plugged into the backplane and the same software is downloaded again. Note: please specify slot 0 when configuring all CPU ports to avoid confusion. Different ladder logic can be loaded in the duty and standby CPUs if required.
- If Ethernet communications are being used, different IP addresses can be used for the Ethernet port of the duty CPU and the Ethernet port of the standby CPU. If the same IP address is used for both CPUs, only the duty mode CPU keeps its Ethernet port active. Both CPUs can keep their Ethernet ports active if they are configured with different IP addresses and #YMODE.4 is set to 1 using ladder logic in the standby CPU. This will allow both CPUs to be polled while connected to the same LAN or a second (redundant) LAN can be used for the standby CPU.
- Install one configured CPU in an even slot address and the other configured CPU in an odd slot address. The even slot CPU will become the duty and the odd slot CPU will become the standby.
- To force a changeover between the redundant CPUs or monitor the changeover status using ladder logic, please see the system register #YEXCEP.
- Note: If using firmware older than 1.41a, a redundancy driver (that matches the firmware version) needs to be downloaded into each CPU before downloading the RTU configuration.

Example

The example below shows how to changeover the CPUs if the duty CPU's real-time clock fails. The ladder logic is also used to display the state of the duty CPU (#R100.8) and the standby CPU (#R100.9).

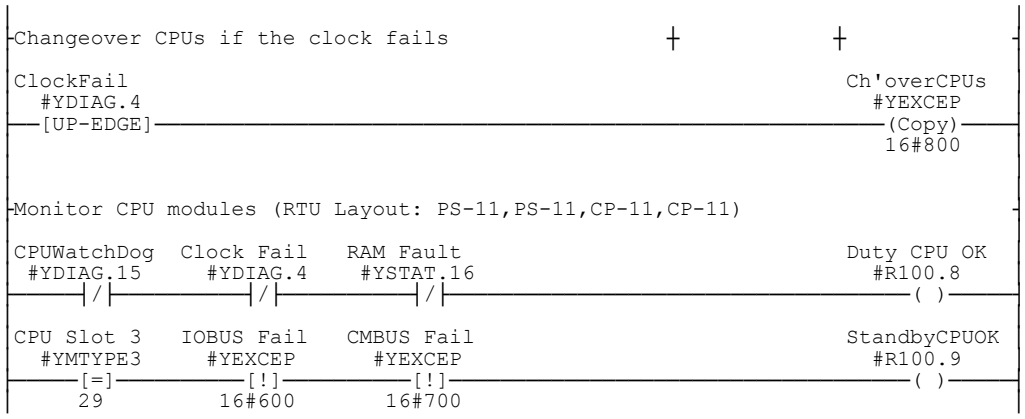


Figure: Redundant CPUs ladder logic

Redundant Power Supplies

Two or more PS-xx power supplies can be plugged into a backplane in any available slots. Each power supply will run normally, sharing the power load (equal load sharing is not guaranteed). If one power supply is removed or fails, the other power supplies will supply the complete power load. Ladder logic is not required when using two of these power supplies on the one backplane.

When two or more power supplies are present on the backplane, one power supply can be hot swapped while the RTU is still running. This does not cause any interruption to the processor or inputs and outputs.

To determine the Total Current supplied by both power supplies to the RTU modules and batteries, the current load for each power supply (#AIss.4) is read and the figures are totalled.

Note: if two or more power supplies are present on the backplane and one is switched off, the Radio, 24V Auxiliary and Battery outputs will still be active on the power supply that is switched off. The Radio and 24V Auxiliary outputs on each power supply can be controlled using ladder logic.

Redundant Communications

It is possible to change the port and/or communications path used to communicate with a remote RTU if there is a communications fail.

The first example below shows how to automatically change which RTU port is used. The second example shows how to automatically change which network path is used.

For both examples the following will happen,

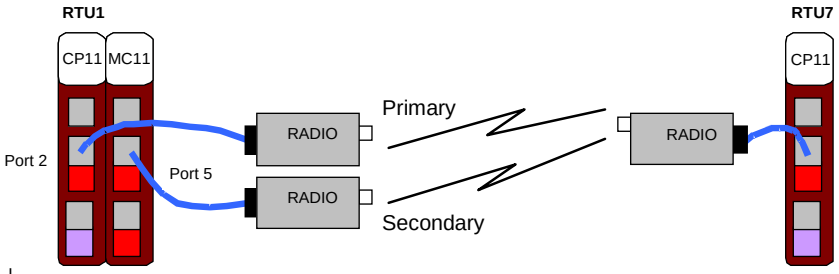
- The RTU initially uses the primary link and if there is a communications failure it switches to the secondary link.
- While using the primary link, the secondary link is tested every 10 minutes.
- When using the secondary link, the primary link is tested every 10 minutes or immediately after the secondary link fails.
- If the primary link is still bad, the network link reverts back to the secondary link.
- If both links fail, the RTU will alternate between the two links every 10 minutes.

#R20.1 is used to show comms fail for the primary link and #R20.2 is used to show comms fail for the secondary link. The comms status for each link is only updated after a message has been sent via that link.

Note: to force a CP-10/11 to close its existing Ethernet socket and use the new network link settings, please see the Network Link register #YLSTrrr.11.

Changing The RTU Communications Port

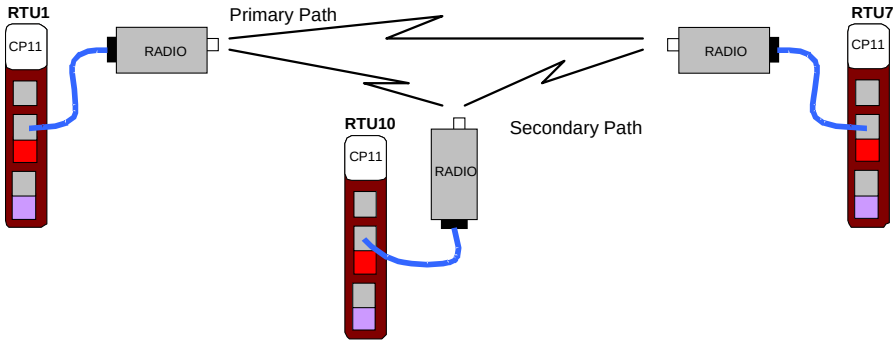
The example below shows how to use CP-xx port 2 as the primary port and MC-11 port 2 (RTU port 5) as the secondary port. The network list is initially configured with a direct network link to RTU7 via port 2.




```
Manage Comms Status

Prim Waiting                                                    Primary Fail
  #YPST2.2                                                        #R20.1
[DOWN-EDGE]-------------------------------------------------------(Copy)
                                                                  #YLST7.1
Sec Waiting                                                     Second Fail
  #YPST5.2                                                        #R20.2
[DOWN-EDGE]-------------------------------------------------------(Copy)
                                                                  #YLST7.1


Test Other Comms Link every 10 minutes or if secondary fails

DoEvery10min                                                    Swap Links
#YTICK.10MIN                                                      #R20.3
 |  |-------------------------------------------------------------(S)
 |  |                                                            |
Sec Waiting  Second Fail                                        Test Comms
  #YPST5.2      #R20.2                                            #R20.4
[DOWN-EDGE]--|  |-------------------------------------------------(S)


Change links if testing, primary fails or prim OK and link = sec

Sec Waiting  Prim Waiting Swap Links   Link=Primary              Swap Links
  #YPST5.2     #YPST2.2      #R20.3      #YLVIA7                  #R20.3
 |/|----------|/|----------|  |---------[=]---------------------(R)
 |                                      2                       |
Prim Waiting Primary Fail                                       Link=Second
  #YPST2.2      #R20.1                                            #YLVIA7
[DOWN-EDGE]--|  |-------------------------------------------------(Copy)
                                                                    5
Sec Waiting  Prim Waiting Swap Links   Link=Second               Swap Links
  #YPST5.2     #YPST2.2      #R20.3      #YLVIA7                  #R20.3
 |/|----------|/|----------|  |---------[=]---------------------(R)
 |                                      5                       |
Prim Waiting Primary Fail Test Comms                            Link=Primary
  #YPST2.2      #R20.1       #R20.4                              #YLVIA7
[DOWN-EDGE]--|/|----------|/|-------------------------------------(Copy)
                                                                    2


If the Test Comms flag is set and ports are free test the link

Test Comms    Sec Waiting  Prim Waiting                         Test Link
   #R20.4       #YPST5.2     #YPST2.2                             RTU   7
 |  |----------|/|----------|/|-------------------------------(RX_DATA)
 |                                                               #R1
 |                                                            Test Comms
 |                                                              #R20.4
 |-------------------------------------------------------------(R)
```

Changing The Comms Path

The example below shows how to swap between a direct comms link to RTU7 (primary) to an indirect comms link via RTU10 (secondary). The network list is initially configured with a direct network link to RTU7 via port 2. Note: to prevent RTU7 "overhearing" indirect messages sent to RTU10, both RTU10 and RTU 7 should have unique System IDs eg. A1 and A2 Hex.

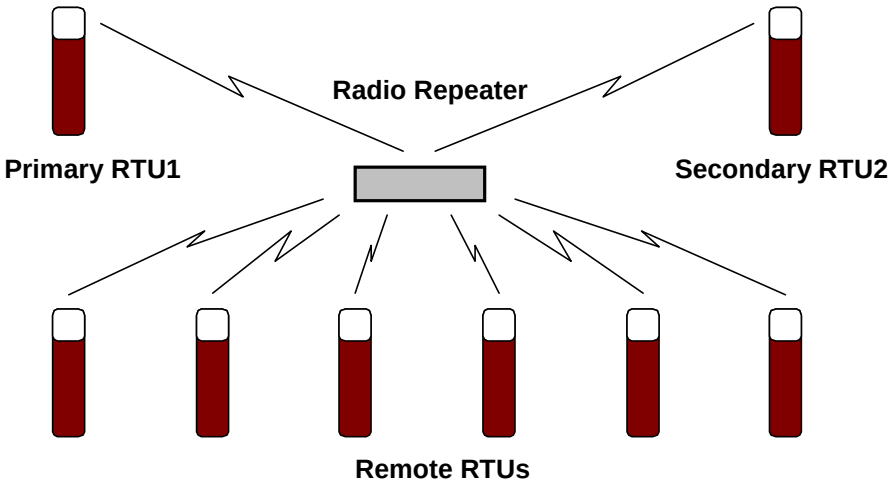



```
Manage Comms Status
MsgWaiting    Link=Primary                                    Primary Fail
  #YPST2.2      #YLDIR7                                         #R20.1
[DOWN-EDGE]-------[=]-------------------------------------------(Copy)-----
                  1                                             #YLST7.1
MsgWaiting    Link=Second                                     Second Fail
  #YPST2.2      #YLDIR7                                         #R20.2
[DOWN-EDGE]-------[=]-------------------------------------------(Copy)-----
                  0                                             #YLST7.1

Test Other Comms Link every 10 minutes or if secondary fails
DoEvery10min                                                  Swap Links
 #YTICK.MIN                                                     #R20.3
-------| |-------------------------------------------------------(S)-------

MsgWaiting    RTU7Link=Sec Second Fail                        Test Comms
  #YPST2.2      #YLDIR7       #R20.2                            #R20.4
[DOWN-EDGE]-------[=]---------| |-----+-------------------------(S)-------
                  0

Change links if testing, primary fails or prim OK and link = sec
MsgWaiting    Swap Links    Link=Primary                      Swap Links
  #YPST2.2      #R20.3        #YLDIR7                         #R20.3
-------|/|---------| |---------[=]-------------------------------(R)-------
                                1
MsgWaiting    Primary Fail Link=Primary                       IndirectLink
  #YPST2.2      #R20.1       #YLDIR7                          #YLDIR7
[DOWN-EDGE]-------| |---------[=]-----+-------------------------(Copy)------
                                1                                 0
                                                              Via RTU10
                                                                #YLVIA7
+           +           +           +                           (Copy)------
                                                                  10
MsgWaiting    Swap Links                Link=Second             Swap Links
  #YPST2.2      #R20.3                    #YLDIR7               #R20.3
-------|/|---------| |-------------------[=]-------------------------(R)-------
                                          0
MsgWaiting    Primary Fail Test Comms                         Direct Link
  #YPST2.2      #R20.1       #R20.4                           #YLDIR7
[DOWN-EDGE]-------|/|---------|/|-------+-------------------------(Copy)------
                                                                  1
                                                              Via Port 2
                                                                #YLVIA7
+           +           +           +                           (Copy)------
                                                                  2
If the Test Comms flag is set and the port is free test the link
Test Comms    MsgWaiting                                      Test Link
   #R20.4       #YPST2.2                                        RTU   7
-------| |---------|/|-------------------------------------------(RX_DATA)----
                                                                  #R1
                                                              Test Comms
                                                                #R20.4
+           +           +           +                           (R)-------
```

Redundant RTUs

It is useful to have a primary and a secondary RTU for a number of reasons:

- All the telemetry system data can be viewed from either RTU
- If the primary RTU fails, the secondary RTU can take control of the system (optional)
- Using completely separate RTUs maximises electrical and physical isolation



A primary master RTU and a secondary master RTU are configured like any other RTUs with unique addresses. The primary RTU polls the outstations and acknowledges exception reports while the secondary RTU simply listens to all the messages from the outstations and updates its own network data. This prevents both master RTUs acknowledging the same message.

Both RTUs need to know the address of the other master RTU (configured in ladder using #Y2NDRTU). Each master RTU is then configured to be in either listen or control mode (configured in ladder using #Y2NDSTAT). When in listen mode, the secondary master RTU updates its network data when it hears new data from an outstation RTU but does not acknowledge or initiate messages. When in control mode, the secondary master RTU acts the same way as the primary master RTU.

There are various ways to configure primary and secondary RTUs with varying complexity. Two different ways are shown below.

Transferring Data Between Two RTUs

To transfer all the data from one RTU to another (including event logs) over a dedicated comms link (eg. modem or Ethernet), the first RTU can use a TX Images block to transfer all its network data, and a TX Update Event Logs block to transfer all its event logs if required. A primary / secondary setup is not required in this instance. The main disadvantage with this arrangement is that data will not be received simultaneously by both RTUs as the second RTU is not able to overhear messages to the first RTU and must wait for the new data to be relayed on. However, data can be updated in the second RTU relatively fast if the first RTU monitors when its network data has changed (using #YLUPDC) and then initiates a TX Images message to the second RTU. New event logs can also be relayed quickly by monitoring when the event log pointer changes (#YLOGIDX) and then initiating a TX Update Event Logs message.

Secondary RTU Always Listens

For this example, the Primary RTU has polling ladder logic while the secondary RTU behaves like an outstation RTU and does not have polling ladder logic.

The example below shows the primary RTU always in control mode and the secondary RTU always in listen mode. This method prevents any clashing of the master RTUs caused when both RTUs think they are in control but does not allow the secondary RTU to take control if the primary RTU fails. If the secondary RTU does not hear from the primary RTU for 35 minutes (waits for a bit longer than 2 polls), the secondary RTU flags a primary RTU comms fail but does not take control. The example includes polling for 2 outstation RTUs and includes a periodic check of the secondary RTU (using the TX IMAGES block) to ensure it has the latest data. The TX IMAGES block checks that the secondary RTU has the same data for RTUs 3 and 4.

When using SCADA software at the primary and secondary RTUs, primary RTU data, setpoints and communication registers need to be transferred by the primary RTU to the secondary RTU in order to view this data at the secondary RTU.

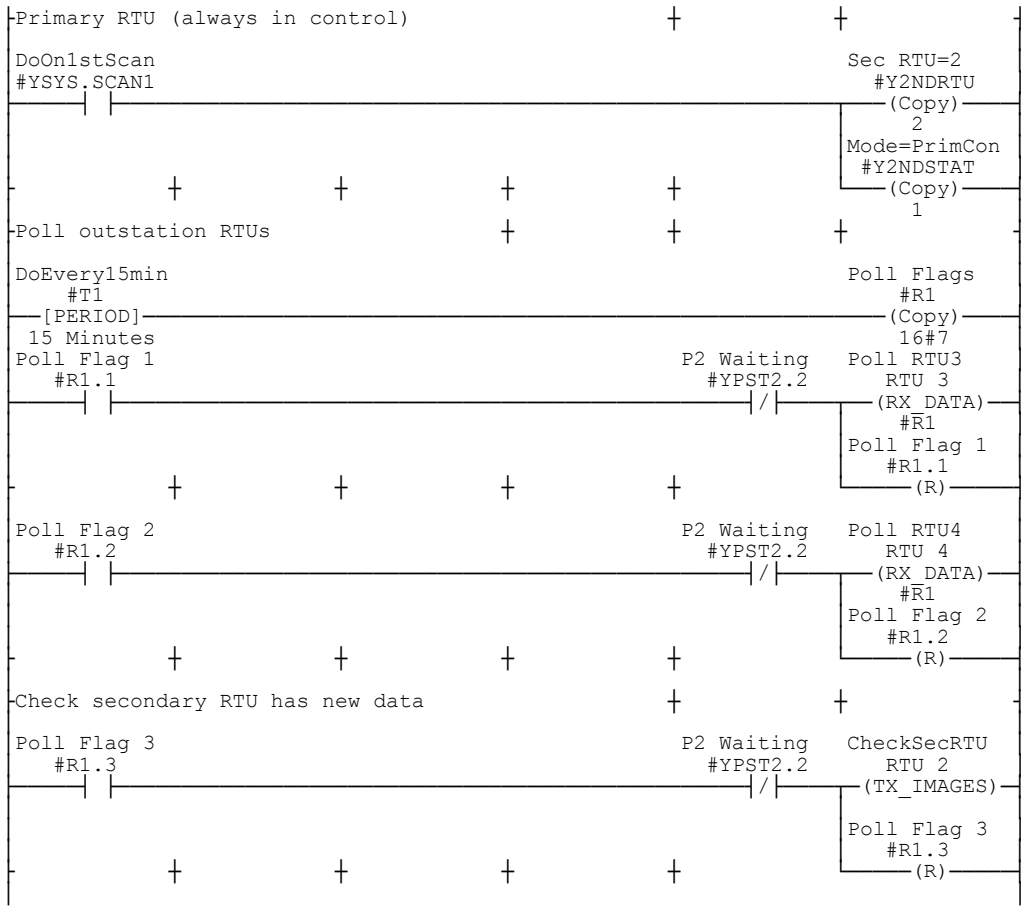


Figure: Primary RTU Always In Control Mode

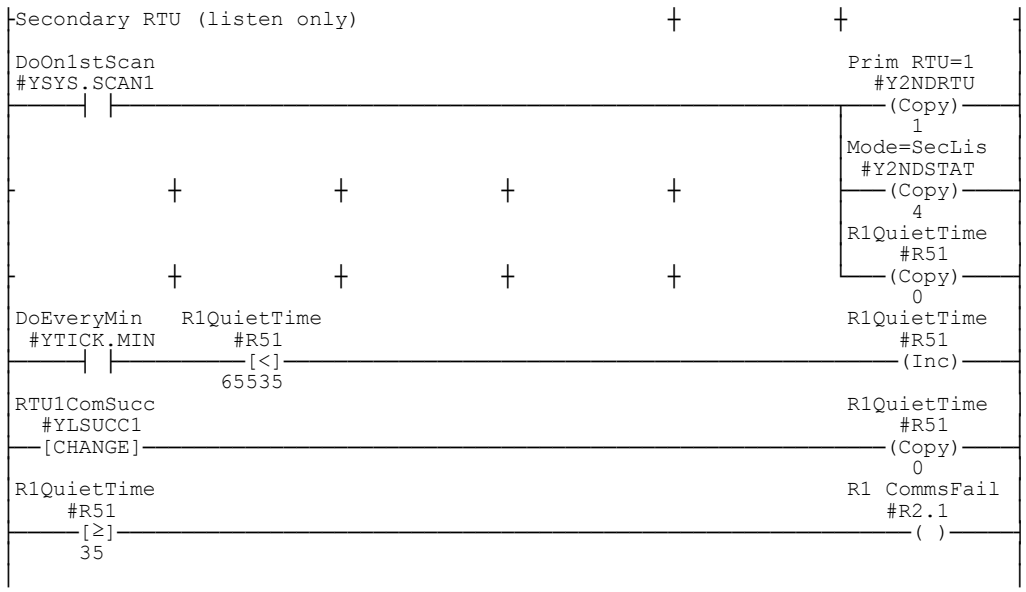


Figure: Secondary RTU Always In Listen Mode

Secondary RTU Takes Control

For this example, the master RTUs use the same ladder logic and are identical except for their RTU address. Edits only need to be made to the primary RTU's ladder logic and then a copy of the ladder logic is used for the secondary RTU.

Primary RTU1 starts up in primary listen mode and then sends a message to secondary RTU2 to make sure it is in secondary listen mode (when RTU2 receives any message from RTU1 it changes to secondary listen mode). If the message is not successful, the primary RTU will keep trying to send a message every minute to secondary RTU2. If the message is successful or a *Force Control* command is received, primary RTU1 changes to primary control mode and begins polling RTU3 and RTU4 every 15 minutes. Secondary RTU2 starts up in Secondary Listen mode and if it has not heard from primary RTU1 for 35 minutes (waits a bit longer than 2 polls), it changes to Secondary Control and begins responding to any messages for primary RTU1 and carrying out polling. When secondary RTU2 is in control mode, it checks if RTU1 is OK at the start of each system poll. If this message is successful, secondary RTU2 immediately reverts to secondary listen mode and will not carry out the system poll.

SCADA Software
The same SCADA software configuration can be used with both the primary and secondary RTUs. The RTU will respond with either its own local data or its network data depending on what RTU address is requested by the SCADA software.

Data common to both masters is read from the primary RTU address (eg. 1). This could include comms success and fail counters for each outstation RTU (stored in local registers in RTU1). When the primary master is active, it updates these counters after any comms successes or fails and then sends these counter values to the secondary master (RTU2). While the secondary master is in listen mode, it overwrites its own comms counters (also stored in local registers) with the network data from RTU1.

If the primary master fails and the secondary master becomes active, the secondary master begins managing its own comms counters. These counters already contain the values last received from RTU1. While the secondary master RTU2 is active, it copies the new value of its own comms counters over the old comms counters stored in network registers for RTU1. This allows the secondary master SCADA software to display the correct value for the comms counters at all times.

Note: some SCADA software drivers (eg. Modbus) allow data to be read from RTU address 0. All RTUs will respond to this address which simplifies some of the data management.

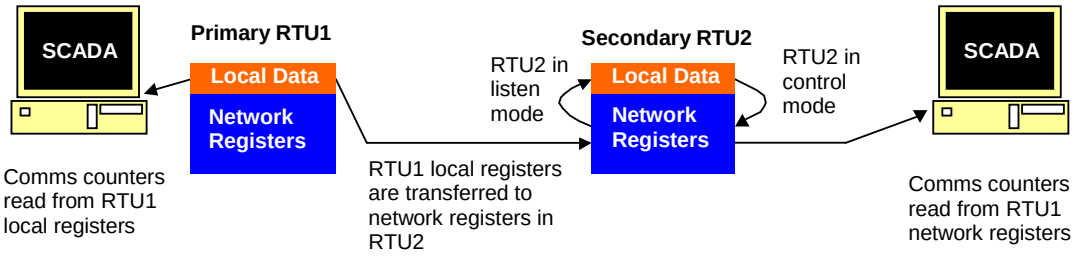


Figure: Managing Common Registers In The Primary And Secondary RTUs

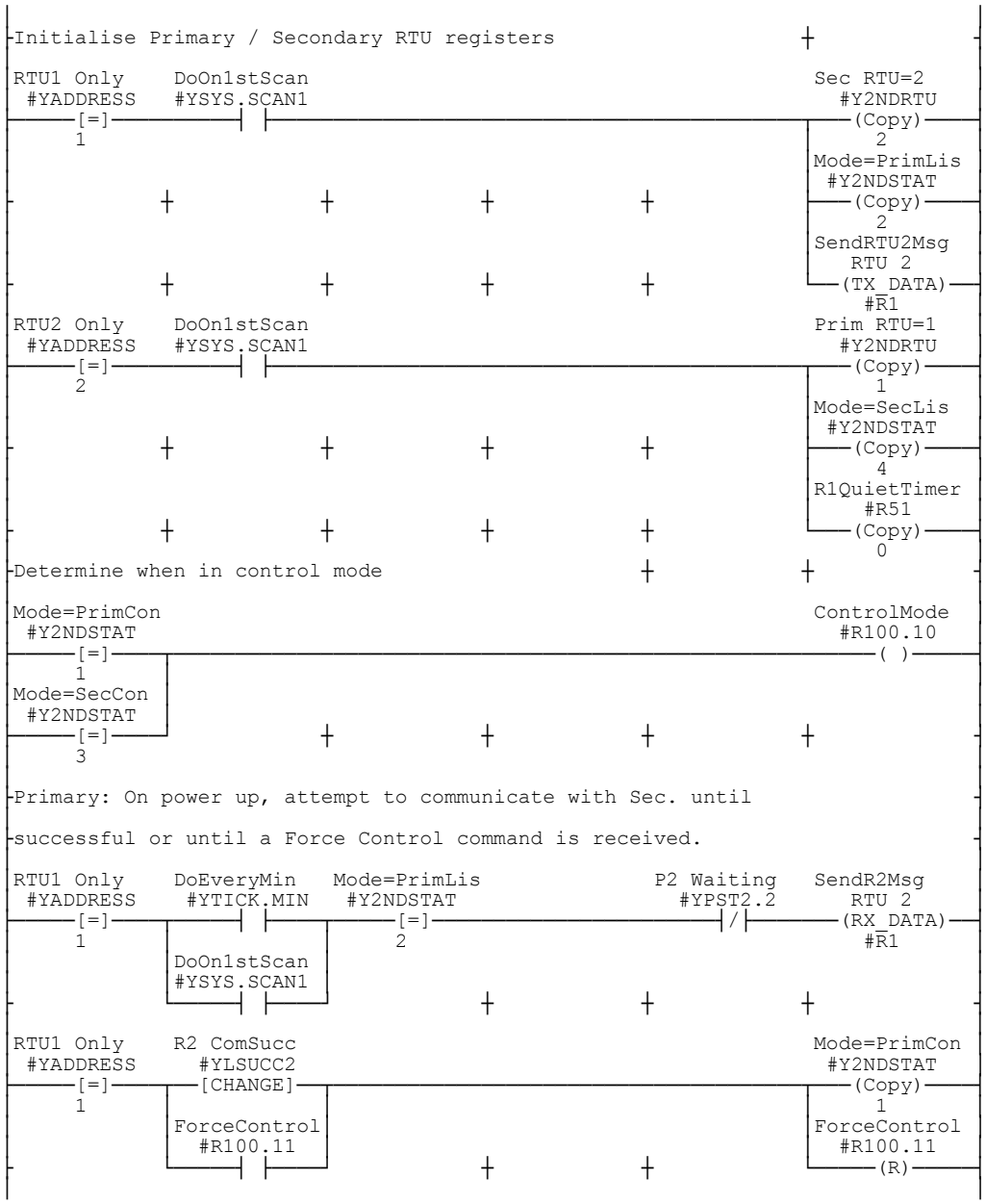


Figure: Primary/Secondary RTU Ladder Logic - PART A

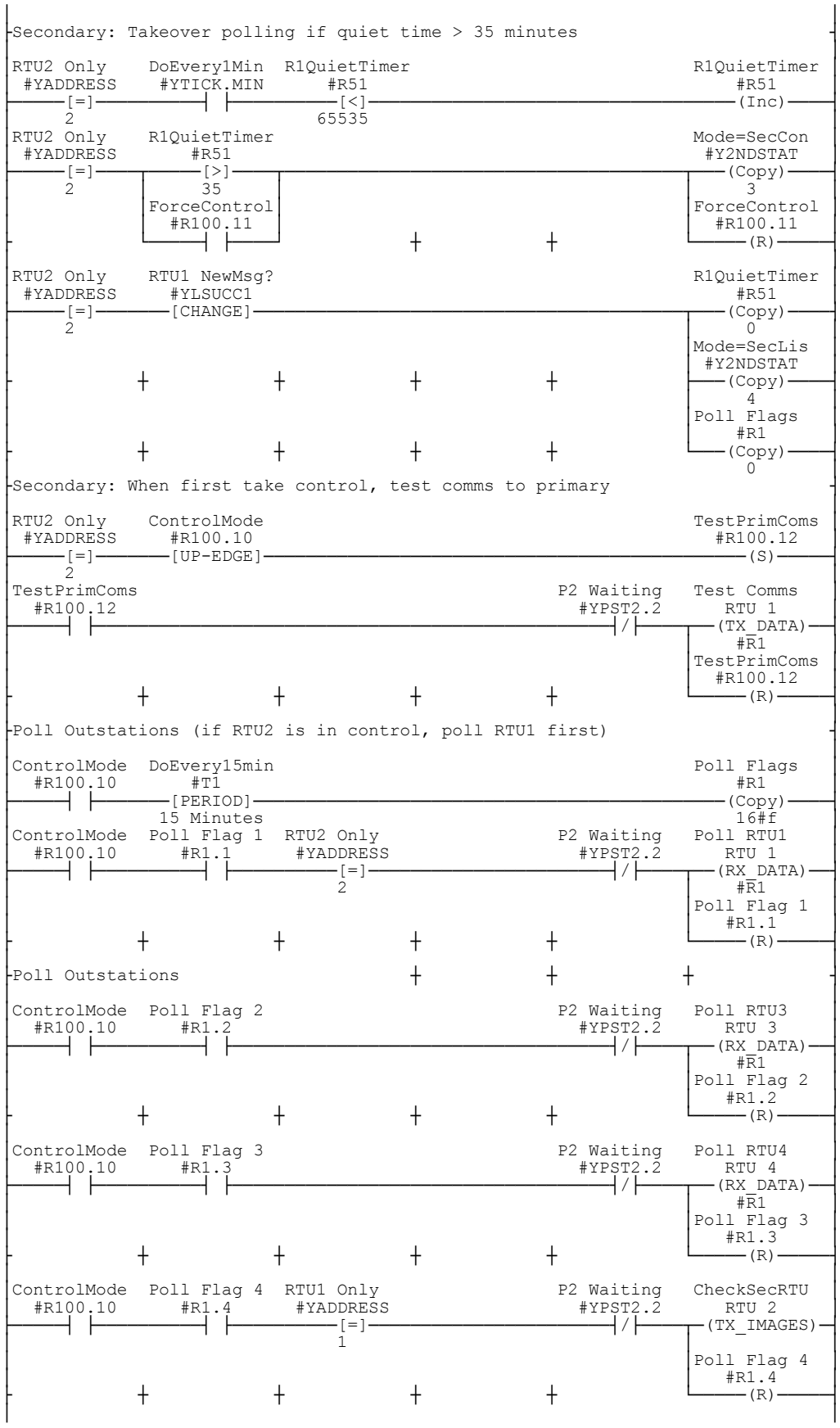
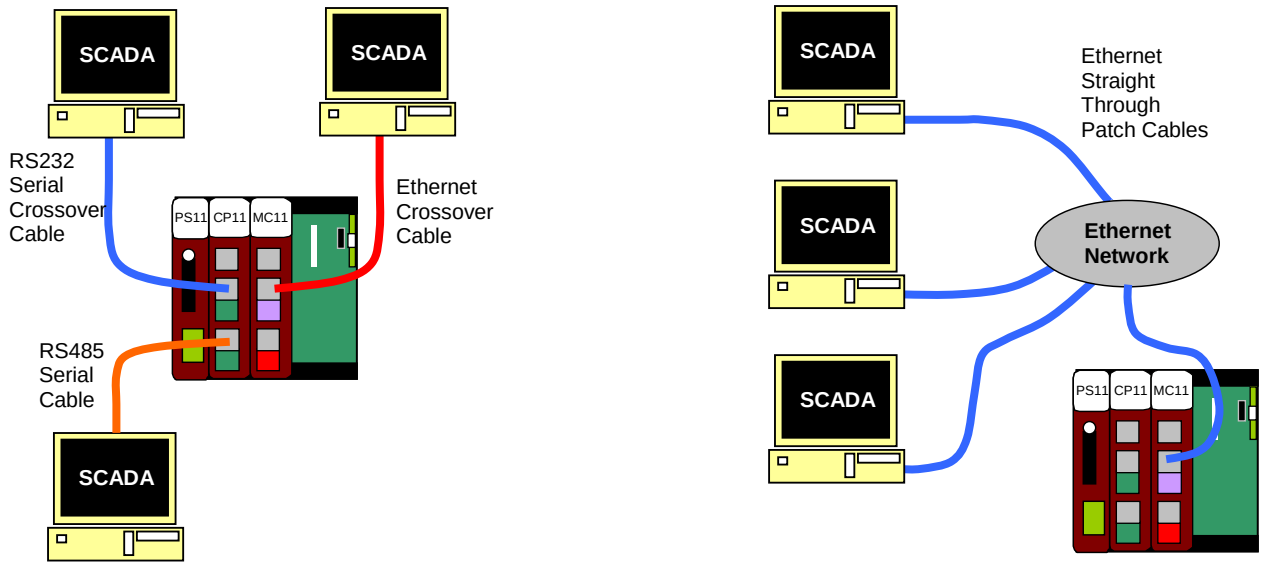

Secondary: Takeover polling if quiet time > 35 minutes
RTU2 Only #YADDRESS [=] 2
DoEvery1Min #YTICK.MIN
R1QuietTimer #R51 [<] 65535
R1QuietTimer #R51 (Inc)
RTU2 Only #YADDRESS [=] 2
R1QuietTimer #R51 [>] 35
ForceControl #R100.11
Mode=SecCon #Y2NDSTAT (Copy) 3
ForceControl #R100.11 (R)
RTU2 Only #YADDRESS [=] 2
RTU1 NewMsg? #YLSUCC1 [CHANGE]
R1QuietTimer #R51 (Copy) 0
Mode=SecLis #Y2NDSTAT (Copy) 4
Poll Flags #R1 (Copy) 0
Secondary: When first take control, test comms to primary
RTU2 Only #YADDRESS [=] 2
ControlMode #R100.10 [UP-EDGE]
TestPrimComs #R100.12 (S)
TestPrimComs #R100.12
P2 Waiting #YPST2.2 /
Test Comms RTU 1 (TX_DATA) #R1
TestPrimComs #R100.12 (R)
Poll Outstations (if RTU2 is in control, poll RTU1 first)
ControlMode #R100.10
DoEvery15min #T1 [PERIOD] 15 Minutes
Poll Flags #R1 (Copy) 16#f
ControlMode #R100.10
Poll Flag 1 #R1.1
RTU2 Only #YADDRESS [=] 2
P2 Waiting #YPST2.2 /
Poll RTU1 RTU 1 (RX_DATA) #R1
Poll Flag 1 #R1.1 (R)
Poll Outstations
ControlMode #R100.10
Poll Flag 2 #R1.2
P2 Waiting #YPST2.2 /
Poll RTU3 RTU 3 (RX_DATA) #R1
Poll Flag 2 #R1.2 (R)
ControlMode #R100.10
Poll Flag 3 #R1.3
P2 Waiting #YPST2.2 /
Poll RTU4 RTU 4 (RX_DATA) #R1
Poll Flag 3 #R1.3 (R)
ControlMode #R100.10
Poll Flag 4 #R1.4
RTU1 Only #YADDRESS [=] 1
P2 Waiting #YPST2.2 /
CheckSecRTU RTU 2 (TX_IMAGES)
Poll Flag 4 #R1.4 (R)

Figure: Primary/Secondary RTU Ladder Logic - PART B

Redundant PCs

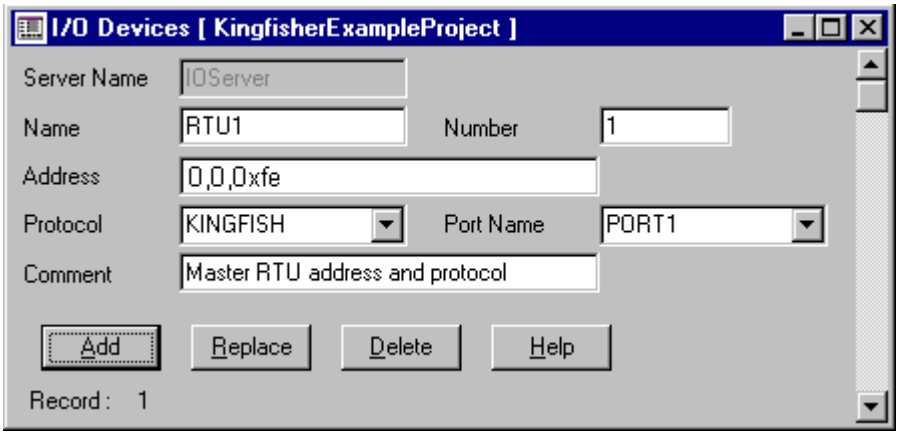
Two or more PCs running SCADA or Toolbox 32 software can be connected to the one RTU as illustrated below. All the PCs can poll the same data and set the same outputs. If one PC fails, the other PCs will continue to operate normally.



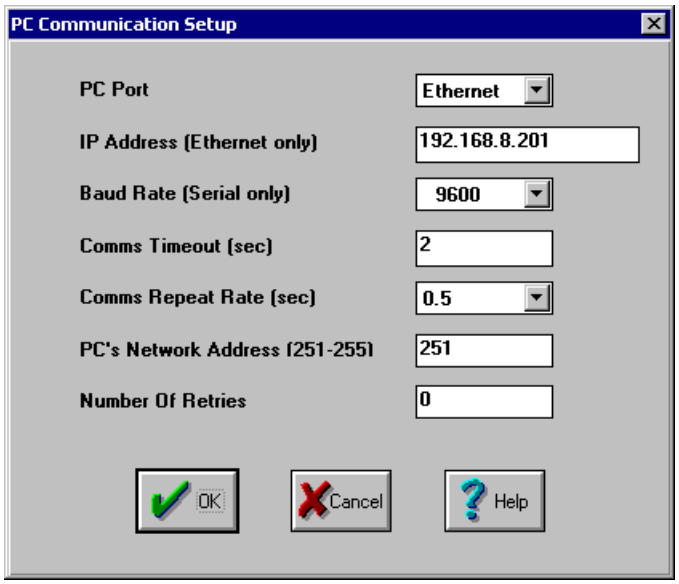
Each PC can be assigned its own RTU port or all the PCs can share one Ethernet port by using an Ethernet Network. Note: a CP-11/MC-11 Ethernet port can handle communications with up to four devices or RTUs simultaneously while using a “T” option Ethernet card or up to eight devices or RTUs simultaneously while using a “T3” option Ethernet card. A CP-12/MC-12 or LP-2/3 Ethernet port can handle communications with up to eight devices or RTUs simultaneously while using a “T3” option Ethernet card

All the PCs can run the same SCADA software configurations with one exception. Each PC must be assigned a unique RTU address in the range 251-255 to prevent communication conflicts.

In the example below, address 254 (**fe** Hex)is assigned to a PC running Citect SCADA software. The address field for the KINGFISH driver is defined as **MasterRTU, ViaRTU, InitiatingRTU**. Address 0 can be used for the Master RTU. Via RTU is usually 0 and **InititatingRTU** is the address of the Citect PC. The KINGFISH driver uses address 255 for Citect by default.



The example below configures Toolbox 32 to use address 251 over an Ethernet network. Toolbox 32 uses address 255 by default.



Appendix - Version Control

Whenever ladder is compiled, Toolbox keeps a check of which functions are used and which versions of firmware are required in order for the ladder logic to run correctly.

After compilation, Toolbox has a record of the oldest and newest firmware versions on which the ladder will run and also has a record of any special firmware drivers that are required. All of this information is then embedded in the header of the compiled ladder file (FILENAME.LLO).

When ladder logic is downloaded into an RTU, Toolbox first requests the RTU Status. The RTU Status provides the firmware version and a list of the firmware drivers that are loaded. Toolbox then checks this against the information stored in the compiled ladder header and if any incompatibilities are found, Toolbox reports the error and prevents the code from being downloaded.

Target Firmware Version
By default, ladder logic will always be compiled to run on the latest (most recent) firmware version. However, some functions in older versions of firmware are incompatible with newer firmware and therefore some ladder compiled for the latest firmware will not run on older firmware. With this menu option it is possible to compile the ladder logic to run on older versions of firmware.

Options currently are:
Version 1.30a or later
Version 1.28a to V1.29f
Version 1.21e to 1.27b
Version 1.21d or before

Changes made in firmware 1.30a
Event logging format was changed. Event log comments are no longer stored. New event logging ladder logic blocks were added.

Changes made in firmware 1.28a
Floating point functions were implemented. The square root function was added (note: the square root function available in previous versions allowed only integer parameters and is incompatible with the new function). An Error handler was implemented for maths exceptions. Previously, it was necessary to compile special code for the Divide and Multiply/Divide functions to ensure that divide-by-zero never happened. A new method of writing to Network Registers was implemented. The number of PID blocks allowed was reduced from 64 to 32. The number of Timer Registers available was increased from 16 to 64.

Changes made in firmware 1.21e
The parameter passing structure (used by the firmware to execute ladder logic) was changed in this version requiring a number of the functions to be compiled differently from older firmware (prior to version 1.21e).

Latest Versions
Text files detailing the features added to each version of Toolbox and firmware are available from http://helpdesk.servelec-semaphore.com.

Appendix - IEC Compliant Register Names

IEC Register Naming Conventions

The IEC-61131.3 standard specifies register names in the following format:

% (type) (size) (address)

type: I = input, Q = output, M = memory
size: X = bit, B = byte (8 bit), W = word (16 bit), D = double word (32 bit), L = long word (64 bit)
address: can be any number of fields, separated by periods (.)

Only Input, Output, and Memory types are allowed in the naming convention. There is no facility for differentiating between Network Registers or System Registers, or any explicit way of differentiating between Analogs and Digitals (other than by size). For greater flexibility an extra address field is used to distinguish between the different Memory register types, and to give them unique IEC 61131.3 compatible names.

IEC Kingfisher Registers

The table below shows the corresponding IEC-compliant name for all register types. For example, #R13 corresponds to %MW13; #DO3.16 corresponds to %QX3.16, etc. (*ch* denotes channel number).

Kingfisher Name	IEC-1131.3 Name	Description
#AI(module).(ch)	%IW(module).(ch)	Analog Input (16 bit)
#AO(module).(ch)	%QW(module).(ch)	Analog Output
#DI(module).(ch)	%IX(module).(ch)	Digital Input (1 bit)
#DO(module).(ch)	%QX(module).(ch)	Digital Output
#DI(module)	%IW(module)	Digital Input (as 16 bit word)
#DO(module)	%QW(module)	Digital Output (as 16 bit word)
#R(number)	%MW1.(number)	Local Register
#R(number).(ch)	%MX1.(number).(ch)	Local Register Bit
#NR(rtu).(number)	%MW2.(rtu).(number)	Network Register
#NR(rtu).(number).(ch)	%MX2.(rtu). (number).(ch)	Network Register Bit
#NA(rtu).(module).(ch)	%MW3.(rtu).(module).(ch)	Network Analog
#ND(rtu).(module).(ch)	%MX4.(rtu).(module).(ch)	Network Digital
#ND(rtu).(module)	%MW4.(rtu).(module)	Network Digital (as 16 bit word)
#T(number)	%MW5.(number)	Timer Register
#Y(sys reg type)	%MW6.(sys reg #)	System Register (Note 1)
#Y(sys reg type).(ch)	%MX6.(sys reg #).(ch)	System Register Bit (Note 1)
#YMTYPE(module)	%MW7.(module).1	System Register : Module Type
#YMVER(module)	%MW7.(module).2	System Register : Module Version
#YP(type)(port #)	%MW8.(port #).(type #)	Port Register (Note 2)
#YP(type)(port #).(ch)	%MW8.(port #).(type #).(ch)	Port Register Bit (Note 2)
#YL(type)(link #)	%MW9.(link #).(type #)	Network Link Register (Note 3)
#YL(type)(link #).(ch)	%MW9.(link #).(type #).(ch)	Network Link Reg. Bit (Note 3)
#Y(type)	%MW10.(type #)	Time/date Register (Note 4)
#YTICK.(type)	%MW10.8.(type #)	Timer Tick Bit (Note 5)

Note 1 : System Register Numbers

1: #YADDRESS
2: #YPRIORITY
3: #YRETRIES
4: #YTIMEOUT
5: #YQUIET
6: #YSYSID
7: #YSTAT
8: #YDIAG
9: #YFIRMW
10: #YIMPL
11: #YEXCEP
27: #YFLAGS
31: #Y2NDRTU
32: #Y2NDSTAT
34: #YRELAYRTU
36: #YMATHSTAT
38: #YPDTIME
39: #YPDSTAT
40: #YPAGERS
41: #YLOGIDX
56: #YMODE

Note 2 : Port Register Numbers

1: #YPMOD
2: #YPADDR
3: #YPTYP
4: #YPSP
5: #YPPRE
6: #YPPOS
7: #YPINAC
8: #YPPCOL
22: #YPST

Note 3 : Network Link Register Numbers

1: #YLDIR
2: #YLVIA
3: #YLTOUT
4: #YLST
5: #YLFC
6: #YLFAIL
7: #YLSUCC
8: #YLSID
11: #YLUPDC
13: #YLLOGIDX

Note 4 : Time/date Registers

1: #YSEC
2: #YMIN
3: #YHOUR
4: #YDAY
5: #YMONTH
6: #YYEAR
7: #YWEEKDAY
8: #YTICK

Note 5 : Timer Tick Numbers

1: 100TH
2: TENTH
3: SEC
4: 10SEC
5: MIN
6: 10MIN
7: HOUR
8: DAY

Appendix - Series I RTUs

Using Series I Communications With A Kingfisher PLUS+ RTU

A Series 2 RTU will respond to Series 1 messages when the port is configured with the S1 option (please see the topic Configuration - Port List, Protocol) and the RTU has the Series 1 driver loaded. A Series 2 RTU will also relay Series 1 messages if the port which the message enters the RTU on is configured as a Series 1 controller (S1 Ctrl). To relay a message from a serial port out of a radio port (eg. to relay an output message from Citect), only the serial port needs to be configured as *S1 Ctrl*. The radio port can be configured as *S1 Outstn*.

Communicating With Series I And Kingfisher PLUS+ RTUs

KIM software is used to configure and communicate with Series 1 RTUs. Since a Series 2 RTU is able to relay Series 1 messages, KIM can be used to communicate through a Series 2 master to Series 1 outstations. It is then possible to carry out the complete range of commands and functions (as if KIM was communicating through a Series 1 master) including downloading a KIM configuration to the outstation RTU.

A Series 2 RTU will respond to KIM's RTU Status command. The KIM RTU Status for a Series 2 RTU only contains the RTU's current time and date with all other fields blank (if the battery voltage is zero, the responding RTU is a Series 2 RTU). A Series 2 RTU will not respond to any other KIM commands such as Get Single or Upload All so in order to read or write data to a Series 2 RTU, Toolbox must be used.

Series 1 RTUs are unable to relay Series 2 messages and so in order to use Toolbox to communicate with a Series 2 outstation, the master RTU must be Series 2 or the PC must be directly connected to the Series 2 outstation.

Converting A Series I Configuration To Kingfisher PLUS+

Ladder Logic

Series I ladder logic can be converted to Series 2 format by loading and saving it with the Toolbox DOS Ladder editor (please contact Semaphore for details). Since Series I only uses 8 digital channels which are stored in the top 8 channels of the 16 channel ID, Toolbox increments all Series I ladder channels by 8. This means that Series I digital channels 1 to 8 are converted to Series 2 digital channels 9 to 16 as this keeps the digital data in the top 8 channels. Note: when creating new Series 2 ladder that is to be used to send digital data to Series I RTUs, digital channels **9 to 16** should only be used as Series I is unable to access digital data in the lower 8 channels.

Analog Inputs

Series I RTUs store analog inputs as a number in the range of 0 to 10,000 while Series 2 store analog inputs as a number in the range of 0 to 32,760. When using a Series I analog input in a Series 2 RTU, care must be taken to clear the bad value flag which is set when the analog input is <4mA (Series I RTUs set Ch16 of the analog input value to ON if the analog input is bad). This must be done before the Series 2 ladder uses the value because Series 2 will treat the bad value as a large value of 32768 (8000H) instead of as a value of zero.

Digital Inputs

Series I RTUs store 8 digital inputs per ID (or register) and these are stored in the MSB of the ID (equivalent to Series 2 digital channels 9 to 16).

Setting Modbus Outputs To A Series I RTU

A Series I RTU will accept Modbus slave messages on port 2 only (CPU3 P2). These Modbus messages can only be used to read and write to the RTU itself as a Series I RTU is unable to relay Modbus outputs to other RTUs. This functionality was designed for local operator interface panels that are directly connected to Series I RTUs.

In order to set Modbus outputs to a Series I outstation through a master RTU, the master RTU must be a Series 2 RTU. The Series 2 master is configured to relay outputs by configuring the port as *S1 Ctrl, Mbus, S2* (please see the topic Configuration - Port List). When relaying outputs to a Series I outstation (an RTU that has a system ID of *AC* in the Configuration - Network List), the Series 2 RTU automatically converts the output to the Series I protocol and then relays the output to the outstation. It is necessary to do this as the output usually goes to the radio port of the Series I RTU (CPU3 P1) which only communicates with the Series I protocol. When relaying digital outputs, the Series 2 RTU sends the digital output in the MSB and the digital mask in the LSB which allows individual channels in the one ID to be set ON or OFF.

When defining the Modbus output addresses for Series I, these are defined the same way as for Series 2 (please see the topic Driver - Modbus). It should be noted that a Series I RTU only has the equivalent of registers 1 to 240 (IDs 1 to 240) and only uses 8 digital channels per register. These 8 digital channels are stored in the MSB or the equivalent of Series 2 channels 9 to 16. Note: outputs to the lower 8 channels of a Series I register (ID) are ignored by the Series 2 master.

The Modbus output address ranges for Analog and Digital Outputs are shown below.

Series I Analog Outputs	
ID	Address
1	41,001
2	41,002
3	41,003
4	41,004
5	41,005
6	41,006
7	41,007
8	41,008
9	41,009
10	41,010
11	41,011
12	41,012
13	41,013
14	41,014
15	41,015
16	41,016
…	
240	41,240

Series I Digital Outputs		
ID	Channel	Address
1	1-8	02,009 - 02,016
2	1-8	02,025 - 02,032
3	1-8	02,041 - 02,048
4	1-8	02,057 - 02,064
5	1-8	02,073 - 02,080
6	1-8	02,089 - 02,096
7	1-8	02,105 - 02,112
8	1-8	02,121 - 02,128
9	1-8	02,137 - 02,144
10	1-8	02,153 - 02,160
11	1-8	02,169 - 02,176
12	1-8	02,185 - 02,192
13	1-8	02,201 - 02,208
14	1-8	02,217 - 02,224
15	1-8	02,233 - 02,240
16	1-8	02,249 - 02,256
…		…
240	1-8	05,833 - 05,840

Sending Series I Pager Messages

Series I pager messages operate the same way as used in Series I systems. The pager message type and target port defined in the Series I pager message are ignored. The settings defined in the Series 2 paging RTU are used instead (this is the *Target Site Address RTU* that is configured in the Series I pager message ID).

Appendix - Calibrating RTU Modules

RT-1 Resistance Temperature Module

The RT-1 module is a resistance/temperature module designed to be used with up to four Pt100 resistance/temperature devices. These devices typically have a resistance of 100 ohms at 0 degrees Celsius but tend to vary slightly in resistance value. The RT-1 module caters for these variances and for the tolerances of its internal components by allowing the user to calibrate the module by setting two different resistance / temperature setpoints. The hardware overview for this module is shown below.

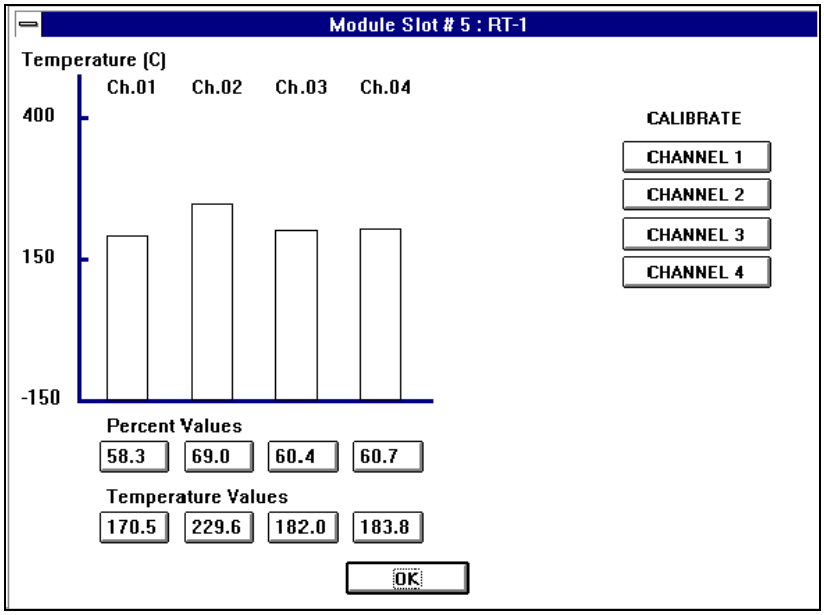


Figure: Hardware Overview For The RT-1 Module

The RT-1 Module operates by scaling the resistance range of the Pt100 to the temperature range of -150 to 400ºC. Once the module is calibrated, it uses its complete 0-100% range (0-32760) to represent the standard temperature range (-150 to 400ºC)

Calibrating The RT-1 Module

Each channel of the RT-1 module can be calibrated by selecting the channel calibration button in the hardware overview. When the channel button is selected the window below is displayed.

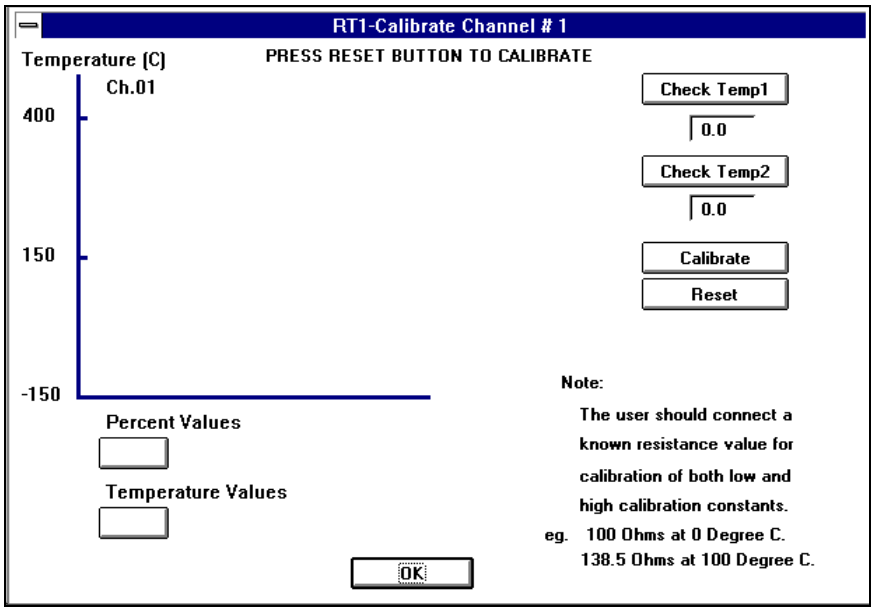


Figure: RT-1 Channel Calibration Window

Calibration Method

- Select the **Reset** button. This will clear the calibration registers in the RT-1 module. If this operation is successful the message "Reset Successfully" will be displayed.
- Connect a resistance to the RT-1 corresponding to the resistance of the Pt100 at a known temperature.
- Select the **Check Temp 1** button.
- Enter the temperature corresponding to this resistance (the allowable range is -150 to 400ºC).
- Connect a resistance to the RTU corresponding to the resistance of the Pt100 at a second known temperature.
- Select the **Check Temp 2** button.
- Enter the temperature corresponding to this resistance (-150 to 400ºC)
- Select the **Calibrate** button. If the setpoints are in range, they will be sent to the RT-1 module and then the message “Calibrate Data Sent Successfully!" will be displayed.

RT-1 Scaling

Say the resistance of a Pt100 device at -150ºC is 40 Ohms and at 400ºC is 250 Ohms. After resetting the RT-1 module, these two resistances might be displayed in Toolbox as 7% and 83% respectively. By entering -150ºC for Temp1 and 400ºC for Temp2, and calibrating the RT-1 module, the 40-250 Ohms resistance range would then be displayed as 0-100% in Toolbox.

When the **Check Temp1** or **Check Temp2** button is selected, Toolbox records the current input percentage. These input percentages are then used with the temperature setpoints to determine an input/temperature characteristic. The figure below illustrates the input/temperature characteristic of a correctly calibrated module.

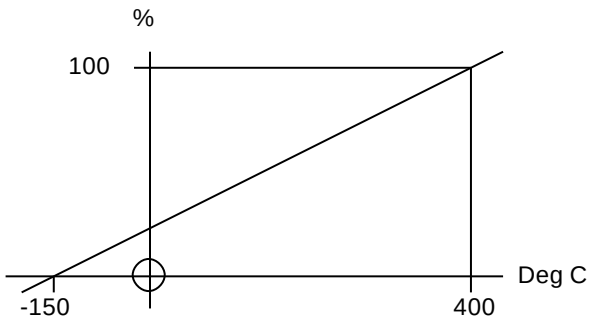


Figure: Output from RT-1 Module After Calibration

RT-1 Out Of Range Calibration Setpoints

Sometimes the values entered for Temp1 and Temp2 and their corresponding input percentages will not allow the RT-1 module to display the complete -150 to 400ºC temperature range. If this occurs the message "Value is out of range!" will be displayed.

For example, after resetting the RT-1 module, Toolbox displayed 100ºC as 40% and 0ºC as 10%. This is illustrated below. In order to cover the complete -150 to 400ºC range, the RT-1 module would need to use the input range of -35 to 130% which it is unable to do as it is limited to 0-100% (0-32760).

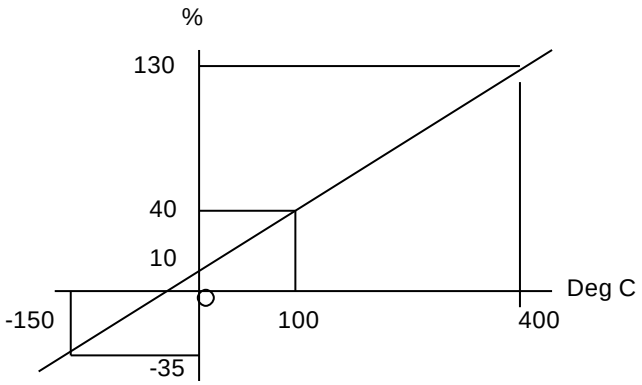


Figure: RT-1 Calibration Setpoints That Are Out Of Range.

Any input/temperature setpoints that will allow the RT-1 module to display -150 to 400ºC within 0-100% will be in range and will allow the RT-1 module to be correctly calibrated. An example of acceptable setpoints are 100ºC=30% and 0ºC=20% and these are shown below.

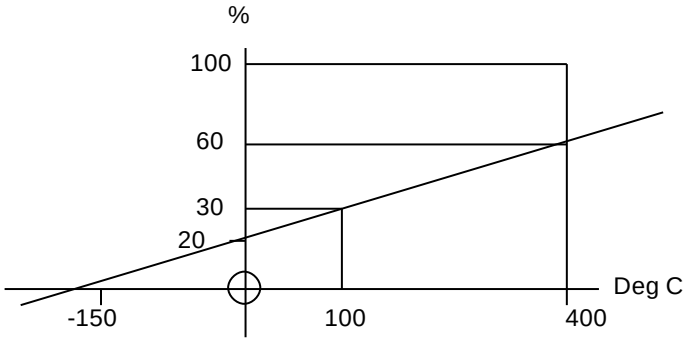


Figure: RT-1 Calibration Setpoints That Are In Range

IO-4 Combination Analog/digital IO Module

The IO-4 module is a combination IO module with 2 analog inputs, 8 digital inputs and 2 digital outputs. The first analog input is designed to be used with a 4-20mA (or 0-20mA) input or with a strain gauge 0-50mV input. When used for a strain gauge input, channel 1 can be calibrated to allow for the different strain gauge characteristics and tolerances of the IO-4's internal components.

There are two hardware versions of the IO-4 module - version 1.2 and version 1.3. The version number is marked on the top of the printed circuit board and is visible through the side of the case. A different Hardware Overview window appears in Toolbox for each version as detailed below.

Note: An IO-4 module is supplied with two 4-20mA analog input channels as standard. To use a strain gauge input on channel 1, the IO-4 must be ordered from Semaphore with this option.

IO-4 Module V1.3

When the **Calibrate Ch. O1** button is selected in the IO-4 hardware overview (**View, Hardware Overview**), additional buttons and fields are displayed as shown below.

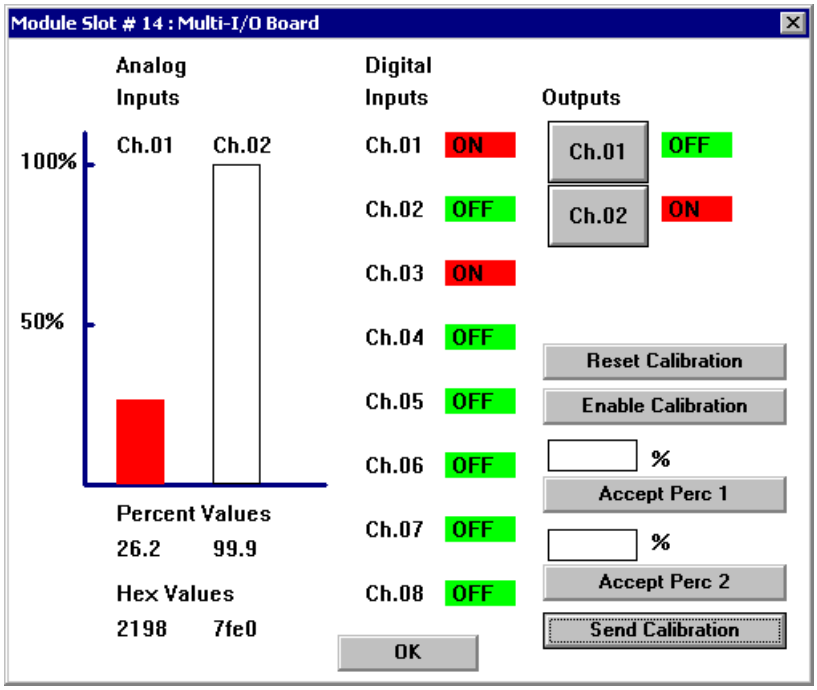


Figure: IO-4 Channel 1 Calibration Window

Calibration Method:

- Select the **Reset Calibration** button. This will clear the calibration registers in the IO-4 module. If this operation is successful, the message "Reset Successfully!" will be displayed.
- Select the **Enable Calibrate** button. This tells the RTU that channel 1 is about to be calibrated.
- Apply the first voltage input to channel 1 (0-50mV).
- Enter the percentage corresponding to this voltage (in the box above the **Accept Perc 1** button) and then select **Accept Perc 1**.
- Apply the second voltage input to channel 1.
- Enter the percentage corresponding to this second voltage (in the box above the **Accept Perc 2** button) and then select **Accept Perc 2**.
- Select the **Send Calibration** button. This will cause the setpoints to be sent to the IO-4 module and then the message "Calibrate Data Sent Successfully!" will be displayed.

Example Calibration

The IO-4 is to be calibrated for a strain gauge input used to measure the water level in a tank. The tank has a depth of 5m. Note: it is more accurate if the strain gauge is calibrated when the tank is full.

- Select the **Reset Calibration** button.
- Select the **Enable Calibrate** button.
- For the first input, lift the strain gauge out of the water (this will probably produce a voltage of about 1mV into the IO-4).
- Enter 0% (corresponding to an empty tank) in the box above the **Accept Perc 1** button and then select **Accept Perc 1**.
- Put the strain gauge back into the water to a known depth of say 2m (this may produce a voltage of say 11mV into the IO-4. The actual voltage will depend on the calibration of the strain gauge). If the tank is full then put the strain gauge all the way into the tank.
- Enter 40% (corresponding to 40% of 5m = 2m or enter 100% if the tank is full) in the box above the **Accept Perc 2** button and then select **Accept Perc 2**.
- Select the **Send Calibration** button. The IO-4 module is now calibrated.

IO-4 Module V1.2

The original IO-4 module (version 1.2) allows channel 1 to be calibrated using fixed percentage setpoints of 0% and 100%. The hardware overview for an IO-4 module, version 1.2 is shown below

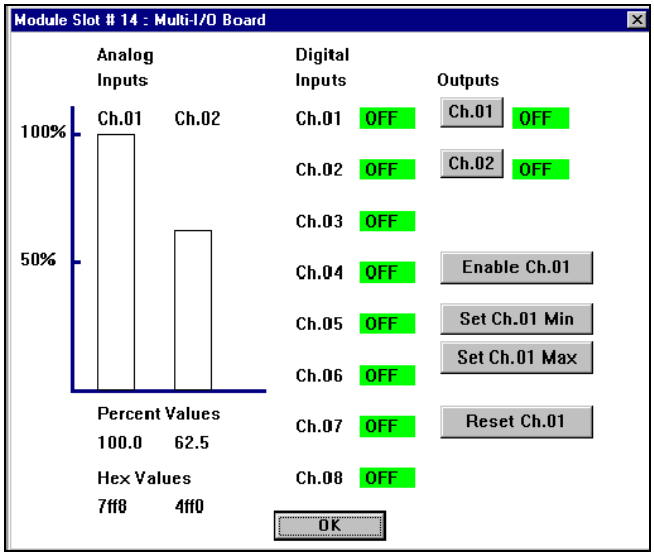


Figure: Hardware Overview For The IO-4 Module V1.2

Calibration Method:

- Select the **Reset Ch.01** button. This will clear the calibration registers in the IO-4 module.
- Select the **Enable Ch.01** button. This tells the RTU that channel 1 is about to be calibrated.
- Apply the first voltage input corresponding to 0% to channel 1 (0-50mV).
- Select **Set Ch.01 Min**.
- Apply the second voltage input corresponding to 100% to channel 1 (0-50mV).
- Select **Set Ch.01 Max**. The IO-4 is now calibrated.

Example Calibration
The IO-4 is to be calibrated for a strain gauge input used to measure the water level in a tank. Note: the tank must be full to allow the IO-4 to be calibrated.

- Select the **Reset Ch.01** button.
- Select the **Enable Ch.01** button.
- For the first input, lift the strain gauge out of the water (this will probably produce a voltage of about 1mV into the IO-4).
- Select **Set Ch.01 Min**.
- Put the strain gauge back into the full tank (this may produce a voltage of say 48mV into the IO-4. The actual voltage will depend on the calibration of the strain gauge).
- Select **Set Ch.01 Max**. The IO-4 is now calibrated.

IO-4 Strain Gauge Test Input

Sometimes it is useful to manually set the strain gauge voltage input and this can be done by using a 0-50mV voltage source as shown below.

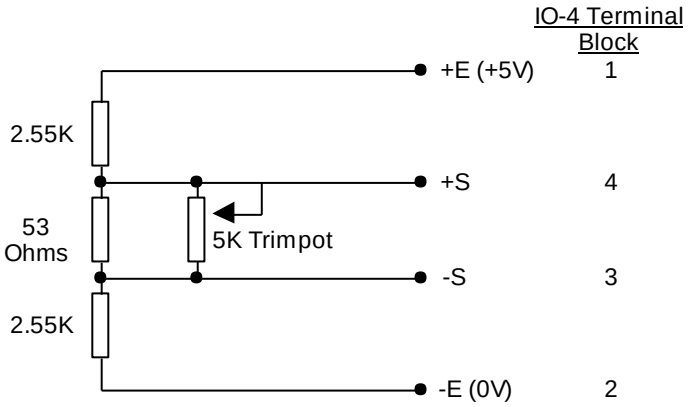


Figure: Circuit for a 0-50mV Strain Gauge Test Device

Appendix - RTU Commissioning

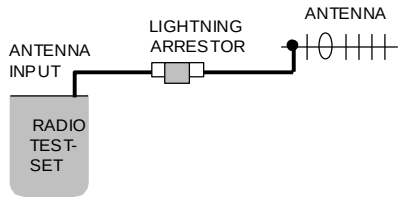
After an RTU has been installed on site, it needs to be made operational. The process of powering up the RTU, downloading software, checking hardware inputs and outputs and establishing communications is referred to as commissioning. The following sections detail the steps involved in the most common installations.

RTU Standard Commissioning

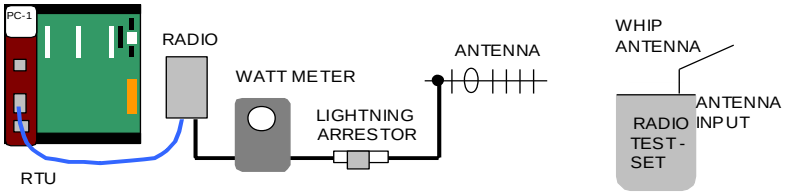
- Check if the correct modules are installed in the correct slots (Refer to RTU Layout Drawings).
- Check backup battery voltage (should be >12V).
- Check that the power supply has been correctly wired (polarity and voltage) and then power up the RTU. Note: if a radio is present, ensure the antenna is properly connected before powering up the RTU and radio.
- For AC powered PC-1 sites, ensure that the PSU-3 voltage is set to 13.8VDC.
- For solar sites, check if the supply voltage from the Solar Regulator is in the desired voltage range.
- Check if the RTU will communicate with Toolbox by viewing the RTU Status.
- Check that all the modules appear in the Hardware Overview and in the correct slots. If the slot numbers are incorrect, the backplane rack switches will probably need to be changed (please see the Kingfisher Hardware Manual or the information written on the backplane).
- Download the RTU configuration and ladder logic (compile the ladder first).
- Check functionality of I/O modules. Simulate a digital input by hard-wiring 12 or 24 VDC to the digital input channel. Check that Digital Output channels can be turned ON and OFF using Toolbox.
- If the RTU includes a PS-1/11, RT-1 or IO-4 strain gauge input, calibrate the module according to the appendix - Calibrating RTU Modules.

Radio Commissioning

- Check that the antenna is correctly mounted and connected to the radio.
- Connect the radio test-set as shown below.

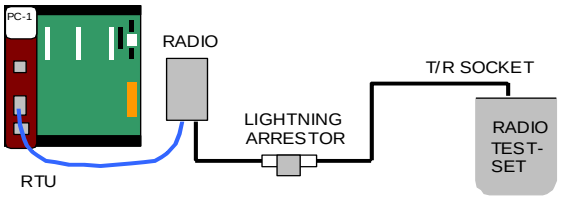


- Measure RX Signal Strength. (This can be done by initially leaving the antenna connected to the radio, performing a 60 second carrier test on the remote RTU using Toolbox **Utilities, Carrier Test**; powering down the local RTU and then connecting the antenna to the test-set as shown above). Ensure that the antenna is pointing in the right direction by using a compass bearing or by using line of sight. Adjust the direction and height of the antenna to get the best signal strength.
- Connect the radio test-set and watt meter as shown below.



- Perform a carrier test on the local RTU by using Toolbox**.**
- Measure forward and reverse power. There should not be any reverse power. Forward power is typically 1 to 5 watts.
- Measure Frequency Deviation and Error. There should be a frequency deviation of 3kHz for wideband radios and 2kHz for narrowband radios. The Frequency Error should be between -1kHz and +1kHz.
- (Optional) Connect the radio test-set and watt meter as shown below.

  Caution! Do not connect the radio cable to the antenna input on the radio test set. Note the transmit power level when the carrier detect LED on the RTU turns ON and OFF.



Private Line Commissioning

A private line is commissioned by establishing communications between 2 RTUs. If the RTUs are unable to communicate, carry out the following steps:

- Telstra sockets and plugs (in Australia) use pins 4 and 6 for private line communications. Ensure that these two wires are being used or that the same pair of wires at each end of the line are being used.
- Kingfisher RTUs use the two outer pins on their private line ports for communications (ie. the top and bottom pins). Ensure that these two outer pins are connected to pins 4 and 6 of the Telstra socket.
- To test whether a signal is getting through, perform a carrier test on the remote RTU using Toolbox (**Utilities, Carrier Test**) and then check to see if 300 to 500 AC millivolts can be measured across the receiving line.
- Ensure the port is configured correctly. Please see the topic Configuration - Port List, Type.

PSTN Modem Commissioning

A modem link is commissioned by establishing communications between 2 RTUs. If the RTUs are unable to communicate reliably, carry out the following steps:

- Telstra sockets and plugs (in Australia) use pins 2 and 6 for PSTN communications. If there is no ring tone, check if these two wires are being used.
- Using the modem directly connected to the PC, use the Toolbox **Utilities, Terminal** program to manually dial the remote site. Check that the modems connect at the correct baud rate and that an RTU status can be obtained. If the modems will not connect, check that the same error correction settings are in both modems (try disabling error correction). If the modems can connect, ensure that they can stay on line for at least 60 seconds by continuously viewing the RTU Status (this will test the line quality).
- Ensure that the local RTU has been configured to wait long enough for the remote RTU to answer (please see the topic Configuration - PSTN, Dial Timeout). For GSM communications, this may take up to 60 seconds.

RTU Troubleshooting

Symptom	Probable Cause	Remedy
Cannot communicate with the RTU	PC not connected to the RTU	Connect the PC/RTU communications cable between the PC and the RTU.
	RTU not powered	Connect power supply and ensure power status LEDs are energised.
	Wrong Toolbox communications port setup	Check the Toolbox communication port parameters in Configuration, PC Setup. These must be the same as the RTU's port parameters. Try View, Auto Detect.
	RTU sleeping or in an unknown state	Power down RTU, remove battery link, wait up to 10 minutes for the RAM to clear and try again.
Ladder Logic will not run	Old version of ladder logic in the RTU	Compile the ladder logic and download it again.
	Ladder logic was not downloaded into the RTU	Compile the ladder and download it into the RTU.
RTU not Powered	No AC/DC and battery flat or not connected	Check the AC or DC source and the battery. If the battery is faulty then replace it with a fully charged battery.
	Power supply installed or wired incorrectly	Check if the PC-1 or PS-xx is plugged firmly into the backplane and wired up correctly. If OK, try a replacement PC-1 or PS-xx.
	AC OK and Battery is flat	Check cable connections to the RTU. Is there a device that continuously drains the battery when the AC and RTU are switched OFF?
PC-1 charge and discharge LEDs are flashing	Battery voltage is at least 0.5V less than the 12V supply input causing more than 1A current to be drawn by the battery.	Irrespective of the charge and discharge LEDs, if the battery voltage is less than the 12V supply, the battery will be charging. Once the battery charges to a level that it draws less than 1A the flicker will disappear.
Analog input readings incorrect	Analog link wrongly configured	Check link is on for 4-20mA and off for 0-20mA.
Analog output readings incorrect	Analog link wrongly configured	Check link is on for 4-20mA and off for 0-20mA.
	Incorrect load	Check that load is correct (250 ohms = 1-5V).
No 24V Aux output	Power saving enabled	Disable RTU's power saving feature.
	RTU not fitted with a 24V converter	Ensure that the RTU is fitted with a converter (optional).
Digital output fuse indicator on	Digital output common incorrectly wired for DC.	Check Hardware Manual or module terminal cover for correct wiring.
One or more modules periodically disappears from hardware overview	Incorrect backplane terminator settings Incorrect I/O Modules List configuration	Check terminator settings and adjust for the required backplane arrangement. Please see the Hardware Manual or backplane labels for details. Fix IO Modules List configuration.

Symptom	Probable Cause	Remedy
RTU loses configuration and ladder logic when powered	Internal battery is flat	Return to Semaphore for replacement of internal battery.
RTU loses configuration and ladder when powered down	Battery link not installed	Fit a link on the back of the processor module. For the PC-1, fit the link across the upper 2 pins.
Modules in wrong slots in hardware overview	Incorrect backplane dip switch settings	Adjust backplane dip switch settings to the correct backplane rack. Turn power off and on again.
Timing is incorrect	RTU Real Time Clock is not set	Set the RTU Real time clock to the current time using Toolbox.
	Battery link is not fitted	Fit a jumper on the back of the processor module. For the PC-1, fit the jumper across the upper 2 pins.
RTU cannot send pager messages using a modem	Modem is not correctly configured	Disable any error correction and compression settings in the modem (ie. use \N0 in the initialisation string).
RTU Will Not Dial / Cannot connect to a remote RTU using a PSTN modem	RTU is not configured correctly / Modem is faulty or incorrectly configured	• Use Comms Terminal to check that the local modem is connected and powered up correctly. Please see Utilities, Comms Terminal for more information. • Check local RTU port is correctly configured. Eg. *Type*=PSTN, *Baud Rate*=9600, *Pre Tx*=0, *Post Tx*=0 (PSTN modem or GSM) or *Post Tx*=400 ms (for 3G modem). Ensure correct *Init String* is being used for the PSTN port (select the **Configure** button). • Check the Network list is correct. Ensure *Timeout*=2000ms (or greater) and that the correct port number (as configured above) is being used to dial the remote RTU. • Check the Phone Directory. Ensure phone numbers are added for the Target (remote) RTU address. • Create a test site - create a Network List entry for Target RTU 100. Then add a phone number of a local phone (eg. a mobile phone) for Target RTU 100 in the Phone Directory. • Test if the local phone will ring, use **Utilities, Dial Site** to dial site 100. • Manually dial the phone number of the remote RTU using a local phone to check that the remote modem will answer. • Check if the local RTU will connect to the remote RTU by using **Utilities, Dial Site** to dial the remote RTU. Ensure the configuration for the remote RTU is loaded in Toolbox. Once connected, select the configuration window of the remote RTU and then select **View, RTU Status**.

Commissioning Site Report

SITE NAME:	
LOCATION:	
DATE:	
COMMISSIONED BY:	

ITEM	**RESULT**
All Kingfisher modules installed with correct options	
All hardware present and connected (eg. modem)	
Fuses installed	
Antenna (if present) connected and properly aligned	
Mains power wired correctly	
DC power wired correctly (correct polarity to battery, radio and 24V Aux supply)	
Battery voltage	V
RTU Powers up correctly. All OK LEDs ON.	
DC Supply Voltages OK (12 and 24V outputs)	
RTU configuration loaded	
RTU ladder logic loaded	
RTU time and date set	
RTU detects all modules (Hardware Overview)	
Radio (if present) RX Level From Master/Sub Master	
Radio (if present) RX Level From Outstation (for store and forward sites)	
Radio (if present) TX Frequency	MHz
Radio (if present) RX Frequency	MHz
Radio (if present) TX Power	W
Radio (if present) TX Reverse Power	W
RTU can communicate with other RTUs	
Site successfully commissioned	

NOTES

Appendix - Ethernet

Overview

- Ethernet can be used on the following RTU ports: CP-21 port 2, CP-11/12 ports 2 and 3 and MC-11/12 ports 2 and 3 (when the MC-11/12 is used with a CP-11/12). It can also be used on LP-2/3 ports 3 and 4.
- A CP-21 uses an **E** option board while a CP-11/MC-11 uses a **T**, **T3 (**with newer firmware revisions) and **A** or **A3** (fibre optic) option board. The CP-12/MC-12 only uses a **T3 or A3** Ethernet Option card. LP-2/3 only uses the **T3 or A3** Ethernet option cards. All Ethernet ports support TCP/IP. CP-11/12 and MC-11/12 Ethernet ports also support UDP/IP.
- CP-11/12 Ethernet ports support the Kingfisher, DNP3, Allen Bradley and Modbus protocols when using the TCP/IP transport layer. MC-11/12 Ethernet ports only support the Kingfisher and Allen Bradley protocols when using the TCP/IP transport layer. For a details of protocols that can be used on Ethernet ports please see the document **protocols.pdf** available from http://helpdesk.servelec-semaphore.com.
- A CP-21 Ethernet port has 24 sockets (19 are available for message transfers) while a CP-11/12 or MC-11/12 Ethernet port has 4 sockets when using a **T or A** Ethernet Option Card and 8 sockets when using a **T3 or A3** Ethernet Option Card.
- The Ethernet port address varies according to the protocol being used.

Protocol	Ethernet Port Address
Kingfisher Series 2	473
Modbus	502
DNP3	20,000
Allen Bradley	2,222
SNMP *	161*

* Only supported by CP-11 Ethernet ports and the TT4030 Tandberg Stream Tracker protocol.

- MAC addresses are assigned to each Ethernet option board as follows:

Option Board	MAC Address
CP-21 E Option	00:01:19:00:00:yy, where: yy = RTU Address as specified in the configuration
CP-11/12 T, A, T3 and A3 Options	00:01:19:00:xx:yy, where: xx = Physical port number of the option board; 01 = Port 2 and 02 = Port 3 yy = RTU Address as specified in the configuration
MC-11/12 T, A, T3 and A3 Options	00:01:19:xx:yy:zz, where: xx = RTU Address as specified in the configuration yy = Physical port number of the option board; 01 = Port 2 and 02 = Port 3 zz = Slot number (in hexadecimal) of the MC module on the backplane; 00 to 3F = Slot 1 to Slot 64
LP-2/3 T3 and A3 Options	00:01:19:00:xx:yy, where: xx = Physical port number of the option board; 02 = Port 3 and 03 = Port 4 yy = RTU Address as specified in the configuration

How The RTU Manages Ethernet Sockets

CP-1x/MC-1x TCP/IP

A CP-11/12 or MC-11/12 Ethernet port has four (eight for **T3**) sockets that can be used for connections. The sockets are used differently depending on which protocol is being used.

- When using the Kingfisher protocol, all four sockets listen on address 473.
- When using the DNP-3 protocol, three (seven for **T3**) sockets listen on address 20,000 for DNP-3 messages and one socket listens on address 473 for Kingfisher messages.
- When using the Modbus protocol, three (seven for T3) sockets listen on address 502 for Modbus messages and one socket listens on address 473 for Kingfisher messages.
- When using the Allen Bradley protocol, three (seven for **T3**) sockets listen on address 2,222 for Allen Bradley messages and one socket listens on address 473 for Kingfisher messages.
- To respond to an incoming message from a new RTU or device, the RTU uses the same socket on which the message arrived.
- A socket that was opened for an incoming message is automatically closed after 60 seconds of inactivity or if the remote device closes the connection or if closed using the **#YLST**rrr**.11** parameter.
- A socket is required when initiating an outgoing message to a new RTU or device. If all sockets are being used, the RTU will disconnect the socket that has been inactive the longest and reuse that socket.

- A socket that was opened for an outgoing message is automatically closed after 60 seconds (by default) or after a configurable amount of seconds as specified by the *Post Tx* port setting.
- When an Ethernet port is re-connected to a network, the Ethernet port will re-initialize all it's TCP/IP connections and send ARP requests if it was a Primary processor port before it was disconnected.
- When all sockets are being used the Ethernet port is unable to respond to messages from new devices.

CP-11/12 or MC-11/12 UDP/IP

CP-11/12 or MC-11/12 Ethernet ports only support the Kingfisher protocol when using the UDP/IP transport layer.

- All sockets listen on address 473 (the destination port).
- UDP messages have a source port and a destination port.
- If **#YPST**rrr**.12** is set to 0 (default), the RTU will respond to a message using a destination port of 473 in the reply. Eg. a message is received with a source port of 2000 and a destination port of 473. The reply is then sent with a source port of 2000+x and a destination port of 473. The reply source port is incremented from 2000 for each newly opened socket.
  All four sockets always remain open listening on port 473 and each socket is able to receive messages from any remote device (provided they are sending to a destination port of 473). However, a separate socket is used when responding to each remote device.
  If all sockets are being used and the RTU needs to initiate a message to a new device, it will disconnect the socket that has been inactive the longest and reuse that socket.
- If **#YPST**rrr**.12** is set to 1, the RTU will respond to a message using the source port of the received message as the destination port in the reply. Eg. a message is received with a source port of 2000 and a destination port of 473. The reply is then sent with a source port of 473 and destination port of 2000. The RTU will continue to use this socket connection to communicate with the remote device until the socket is reused by the RTU for a new connection.
  If all sockets are being used and the RTU needs to initiate a message to a new device, it will disconnect the socket that has been inactive the longest and reuse that socket.
- UDP socket connections do not time out.
- UDP socket connections cannot be closed by the remote device.
- An RTU will only use the reply IP address/port numbers to communicate with a remote RTU while the failed attempts is less than three. After three failed attempts, the socket is closed and available for re-use.

CP-21 TCP/IP

A CP-21 can have up to 24 socket connections at the same time. Some sockets are reserved for listening for incoming messages for individual protocols as follows:

- 1 socket always listens on port 473 (for Series 2 connection requests)
- 1 socket always listens on port 20000 (for DNP3 connection requests)
- 1 socket always listens on port 502 (for Modbus connection requests)
- 1 socket always listens on port 80 (for the HTTP server - no longer supported)
- 1 socket always reserved for creating new socket connections.
- The remaining 19 sockets are available for data connections to remote devices. When an incoming message is received by one of the 4 listening sockets then a new connection is made using one of the 19 sockets to reply. The socket used for the incoming connection will not automatically time out.
- When all 19 sockets are being used the Ethernet port is unable to respond to messages from new devices.
- When all 19 sockets are being used and the local RTU needs to initiate a message to a new RTU that doesn't already have a connection, the RTU will close the socket that has been inactive the longest and reuse that socket.
- Sockets can be closed by the remote device.
- A socket used for a connection that was initiated by the *local* RTU can be closed after a configurable amount of seconds by using the *Post TX* port setting (as detailed below).

LP-2/3 TCP/IP

An LP-2/3 Ethernet port has eight sockets that can be used for connections. The sockets are used differently depending on which protocol is being used.

- When using the Kingfisher protocol, all eight sockets listen on address 473.
- When using the DNP-3 protocol, seven sockets listen on address 20,000 for DNP-3 messages and one socket listens on address 473 for Kingfisher messages.
- To respond to an incoming message from a new RTU or device, the RTU uses the same socket on which the message arrived.
- A socket that was opened for an incoming message is automatically closed after 60 seconds of inactivity or if the remote device closes the connection or if closed using the **#YLST**rrr**.11** parameter.
- A socket is required when initiating an outgoing message to a new RTU or device. If all sockets are being used, the RTU will disconnect the socket that has been inactive the longest and reuse that socket.

- A socket that was opened for an outgoing message is automatically closed after 60 seconds (by default) or after a configurable amount of seconds as specified by the *Post Tx* port setting.
- When an Ethernet port is re-connected to a network, the Ethernet port will re-initialize all it's TCP/IP connections and send ARP requests if it was a Primary processor port before it was disconnected.
- When all sockets are being used the Ethernet port is unable to respond to messages from new devices.
- If all sockets are being used and the RTU needs to initiate a message to a new device, it will disconnect the socket that has been inactive the longest and reuse that socket.

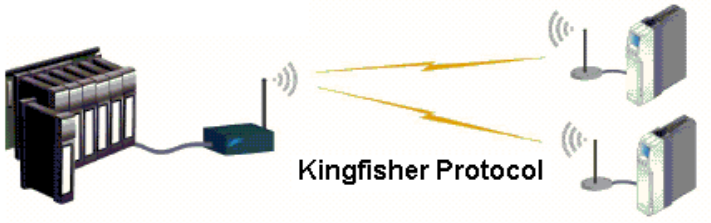
Ethernet Configuration

The following parameters can be used to control the operation of the RTU's Ethernet ports:

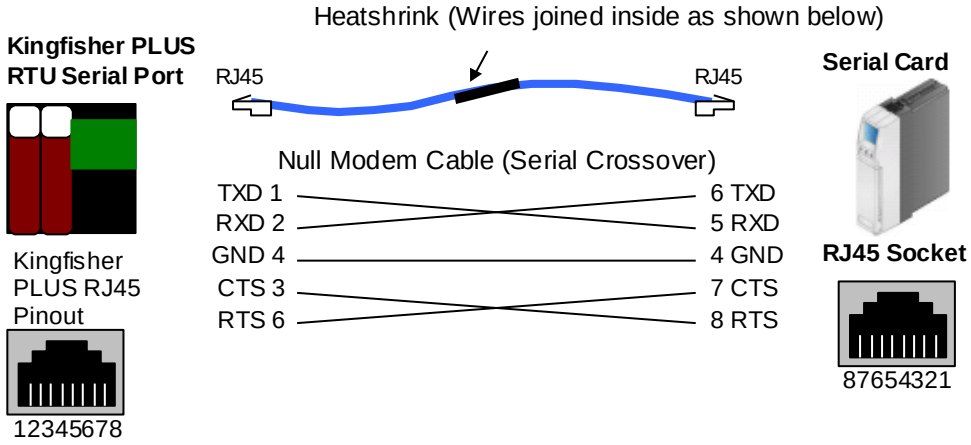
<table>
<tr><td>#YPSTnn.7
(nn=port number 1 to 16)</td><td colspan="4">1 = Link Active (CP-1x or MC-1x ports only). Note: supported by Ethernet T option boards version 1.1 and newer (port is labeled E'NET-T 1.1)</td><td>Read only</td></tr>
<tr><td>#YPSTnn.12
(nn=port number 1 to 16)</td><td colspan="4">0 = Use fixed UDP destination port based on port protocol [default]
1 = Use dynamic UDP destination port. Required when using Citect 6.x+.
Note: this parameter is not supported by MC-10/11/12 Ethernet ports</td><td>Read/Write</td></tr>
<tr><td>Post Tx
(configured for a port from Configuration, Port List)</td><td colspan="4">The inactivity timeout for each Ethernet socket. Once the socket has been inactive for this time setting, the socket will be closed. Implemented in CP-10/11/12 firmware 1.43d, CP-21 firmware 1.45e and MC-11/12 firmware C153 or newer. Also LP-2/3 firmware 1.50e
CP-21 Ethernet | CP-10/11/12, MC-10/11/12, LP-2/3 Ethernet
Post Tx | Socket Timeout (s) | Post Tx | Socket Timeout (s)
0 | 60,000 [default] | 0 | 60 [default]
1 to 9 | 10 | 1 to 600 | 1 to 600
10 to 600 | 10 to 600 | > 600 | 600
> 600 | 600 | |</td><td>Read/Write</td></tr>
<tr><td>#YLSTrrr.11
(rrr=remote RTU address)</td><td colspan="4">1 = Close socket connection (CP-1/121 or MC-11/12 ports only). When set, forces the RTU to close the existing socket connection for a remote RTU. When a new message is initiated, the new network link settings will then be used (useful for creating a redundant comms path).</td><td>Read/Write</td></tr>
<tr><td>#YMODE.4</td><td colspan="4">1= Keep Ethernet port open on the redundant standby CPU. The duty and standby CPUs can both have their Ethernet port working simultaneously as long as they are configured with different Ethernet addresses (allows support for redundant LANs).</td><td>Read/Write</td></tr>
<tr><td>#YSOCKETx
(x=socket number 1 to 24)</td><td colspan="4">(CP-21 only) Returns the network usage of the 24 possible CP-21 Ethernet sockets on the LAN. 0=not used, 1-255=the address of the RTU using that socket.</td><td>Read only</td></tr>
<tr><td>#YLIPADLrrr
(rrr=remote RTU address)</td><td colspan="4">(0-65535) Lower half of IP address A.B.C.D. Chs 1-8=C, Chs 9-16=D. To set C and D, use #YLIPADLrrr=C+256xD.</td><td>Read/Write</td></tr>
<tr><td>#YLIPADHrrr
(rrr=remote RTU address)</td><td colspan="4">(0-65535) Higher half of IP address A.B.C.D. Chs 1-8=A, Chs 9-16=B. To set A and B, use #YPIPADHrrr=A+256xB.</td><td>Read/Write</td></tr>
</table>

Appendix - Communicating With A G3 RTU

If the G3 is setup as a *Remote IO* device it can then communicate with a Kingfisher PLUS+ RTU using a subset of the Kingfisher protocol. Remote IO mode allows exception reports to be sent by a G3 to a Kingfisher RTU and allows outputs to be written to a G3.



Serial Test Cable



Kingfisher PLUS+ RTU Setup

- **Configuration, Address & Description:** Set the RTU's *Site Address* in the range 1 to 249. The RTU's address must be different to the G3's address.
- **Configuration, Port List:** Set port *Type* to RS-232 if using a serial cable, Trio M Series Radio, MaxStream external spread spectrum radio or Spread Spectrum option board. Set port *Type* to PSTN if using a dialup modem or GSM. If using a Trio M Series radio, ensure *Baud Rate* is set to match the radio speed. If using a spread spectrum radio option board, set *Baud Rate* to 9600 (for 9XTend US / Australian radio) or 19200 (for 24XStream International radio). For all setups, *Pre TX* and *Post TX* can be left set to 0 ms and *Protocol* can be left set to Series 2.
- **Configuration, Network List:** add a link with the G3 device address and the port number of the communications device as configured above.
- **Configuration, Phone Directory:** if using a dialup modem or GSM, configure a phone number corresponding to the G3 address.
- If using a spread spectrum radio (option board or external), please initialise it with the settings detailed below.

MaxStream External Spread Spectrum Radio

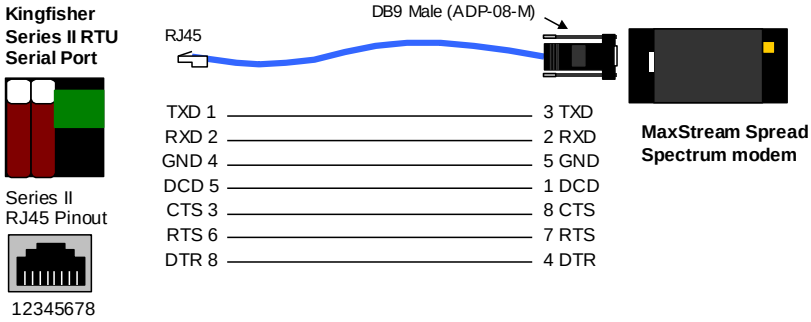
Part Numbers

The following Digi (formerly MaxStream) external spread spectrum RF modems are recommended:

- XTH9-PKI-R-128 Xtend 900MHz Spread Spectrum Modem RS232/485/422 for Australia
- XT09-PKI-R Xtend 900MHz Spread Spectrum Modem RS232/485/422 for the USA
- X24-019PKI-R Xstream 2.4GHz Spread Spectrum Modem RS232/485/422, 19200 bps throughput, International

Check for your nearest Digi supplier at www.digi.com.

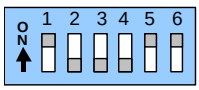
Interface Cable



Radio Switch Settings

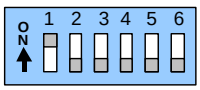
Please ensure that the MaxStream radio switches are set as follows:

XTend



Set for: RS232,
RS485 termination=None and
TX/RX mode=User Defined
(Switch 1, 5 and 6 set ON)

XStream



Set for: RS232,
RS485 termination=None
and Parity=None
(Switch 1 ON)

Software Settings – Using MaxStream X-CTU Software

Note: radios in a network can only communicate if they have the same settings for Vendor ID and Hopping Channel.

- Connect a powered up MaxStream radio to your PC's COM port (eg COM1).
- Run the MaxStream X-CTU program supplied with the radio.
- From the **PC Settings** tab, ensure X-CTU is setup to use the right COM port, 9600 baud (XTend) or 19200 baud (XStream), 8 data bits, no parity and 1 stop bit.
- Select the **Test / Query** button to check communications to the radio.
- From the **Modem Configuration** tab, select the **Read** button. All the radio settings will be read into the program.
- From the **Networking & Security** section of the radio settings:
  Set **Hopping Channel** to the same number as all the other radios in your network.
  Set **Destination Address** to **FFFF** Hex.
  Set **Source Address** to a unique value in your network (all radios in the network should have a different Source Address in the range 0-FFFF Hex).
  Set **RF Mode** to **Repeater** or **Repeater End Node (**Repeater is used to relay messages to another radio and will also allow the local radio to communicate).
  Note: **Modem VID** (**Vendor ID)** should be set to the same number as all the other radios in your network. This can be set using a terminal program as described below.
- If using an XTend radio, scroll down to the **RF Interfacing** section of the radio settings and change BR to 0 (ie. set RF data rate to 9600 bps).
- Select the **Write** button. The settings will now be written and saved in the radio.

Software Settings – Using A Standard Terminal Program *

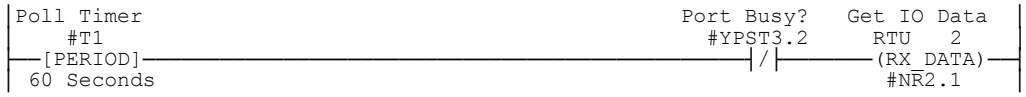
Radio parameters can be set using Windows HyperTerminal (or the Terminal window in the MaxStream X-CTU software). Once connected to the radio, type +++ to put the radio in command mode. Then enter the relevant AT commands below. Note: radios in a network can only communicate if they have the same settings for Vendor ID and Hopping Channel.

+++	Radio responds with OK when ready to receive commands
ATRE <enter>	Reset the radio to factory defaults
ATHP0 <enter>	Set Hopping Channel to 0 (default) or in the range XTend 0-9, XStream 0-6
ATDTFFFF <enter>	Set Destination Address to FFFF Hex.
ATMY15 <enter>	Set Source Address to eg. 15 (must be unique) or in the range 0-FFFF Hex
ATID3332 <enter>	Set Vendor ID to 3332 Hex (default) or in the range XTend 0-7FFF Hex, XStream 10-7FFF Hex.
ATMD5 <enter>	Set RF Mode For XTend: 5=Repeater, 6=End Node. For XStream: 3=Repeater, 4=End Node.
ATBR0 <enter>	Only required if using an XTend radio. Sets RF data rate to 9600 baud.
ATWR <enter>	Save settings to permanent memory

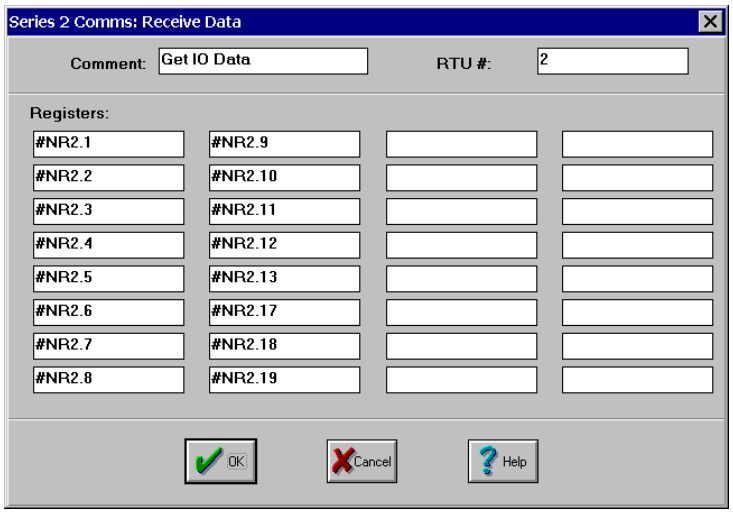
* When configuring a Kingfisher PLUS RTU (CP-11/12) Spread Spectrum Option card, the Toolbox 32 Comms Terminal is used. Setup the Comms Terminal to use the Remote RTU Port Number that corresponds to the option card and then enter the above settings.

Polling a G3 Using A Kingfisher PLUS+ RTU

Configure ladder logic to poll *network* registers from the G3 using the RX DATA block (an example is shown below)



The above example polls the G3 (address 2) every 60 seconds, provided that port 3 is not busy. The RX Data block was setup as shown below.



Exception Reporting from a G3 to a Kingfisher PLUS+ RTU

When a G3 exception reports it sends the first 12 registers as detailed below.

G3 Kingfisher Registers

Register *	Description
#NRrrr.1	Ch 1 (LSB) = Digital input 1 status Ch 2 = Digital input 2 status
#NRrrr.2	Analog input 1. 0-26208 = 0-100% = 0-20mA or 4-20mA For over-range: 0-20mA input: 0-32760 = 0-125% = 0-25mA 4-20mA input: 0-34398 = 0-131.25% = 4-25mA
#NRrrr.3	Analog input 2. Same range as above.
#NRrrr.4	General alarms Ch 1 ON = Comms Fail Ch 2 ON = Low Voltage alarm Ch 3 ON = Incorrect PIN alarm
#NRrrr.5	Supply Voltage to the G3 (0 to 300 = 0.0 to 30.0V)
#NRrrr.6	SS radio received signal strength. There are two ranges depending on which type of radio is installed. 0= no signal received **9XTend radio (900MHz):** -110 (weak) to -40 (strong) dBm **24XStream radio (2.4GHz):** 6 (weak) to 54 (strong) The 9XTend signal strength is stored as 65426 to 65496 (unsigned) = -110 to -40 (signed) dBm.
#NRrrr.7	G3 internal temperature. -20 to 70°C (already scaled)
#NRrrr.8	Status indications Ch 1 ON = IO Scanning ON Ch 2 ON = if G3 in override mode, IO data has changed. Ch 3 ON = log buffer is ≥90% full (=900 logs) Ch 4 ON = new counter 1 value Ch 5 ON = new counter 2 value (Remote G3 sends counter value and then resets it back to zero)
#NRrrr.9 and #NRrrr.10	Counter 1. 32-bit long integer (1 to 4,294,967,296)
#NRrrr.11 and #NRrrr.12	Counter 2. 32-bit long integer (1 to 4,294,967,296)
#NRrrr.13	Counter periods. Used by remote G3 for frequency generator outputs. Ch 1 - 15: Index into counter table Ch 16 ON = Use table (ms): 500, 1000, 5000, 30 000, 60 000, 90 000, 180 000, 360 000, 86 400 000 Ch 16 OFF = Use table (s): 0, 60, 3600, 21 600, 43 200, 604 800
#NRrrr.14	Ch 1 = DO1 raw output state – Read/Write [#] Ch 2 = DO2 raw output state – Read/Write [#]
#NRrrr.15	AO1 raw output value. 0-26208=0-100% - Read/Write [#]
#NRrrr.16	AO2 raw output value. 0-26208=0-100% - Read/Write [#]
#NRrrr.17	Ch 1 = DO1 actual output state - Read only Ch 2 = DO2 actual output state - Read only
#NRrrr.18	AO1 actual output value. 0-26208=0-100% - Read only
#NRrrr.19	AO2 actual output value. 0-26208=0-100% - Read only

* rrr = G3 address

[#] The raw outputs are modified by the G3 output options before the actual output is set.

Setting G3 Outputs Using a Kingfisher PLUS+ RTU

The following *network* registers can be sent to a G3 to set the outputs:

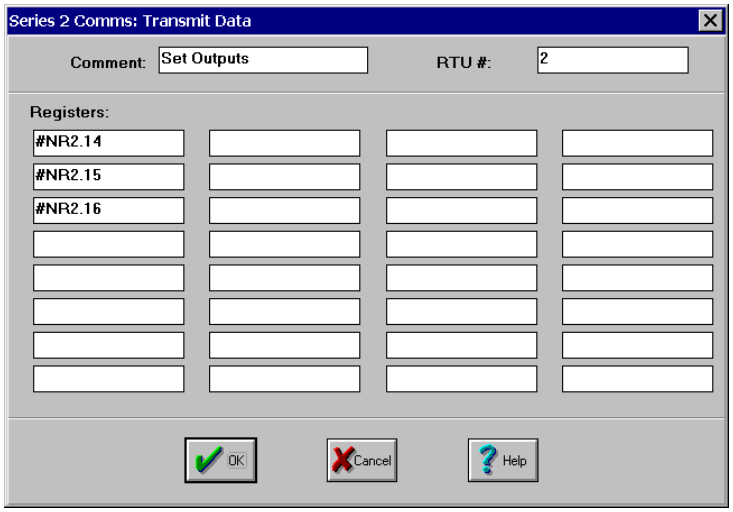
#NRrrr.14 Ch 1 = DO1 output state
Ch 2 = DO2 output state
#NRrrr.15 AO1 output value (0-26208=0-100%)
#NRrrr.16 AO2 output value (0-26208=0-100%)

Where *rrr* is the G3 address to set the outputs in.

Notes:

- Only *network* registers (eg #NR2.14) can be sent to the G3. *Local* registers (eg #R14) will not be accepted.
- To set outputs in a G3, first set the network register(s) in the Kingfisher PLUS RTU and then send the network register(s) to the G3 using a TX DATA block.
- When sending a digital output, the *whole* register is sent to the G3 (eg #NRrrr.14) and not just a single bit (eg #NRrrr.14.1).
- The G3 applies any configured output options and events to all new outputs.

An example TX Data block that will write to all the G3 outputs (address 2) is shown below.



Polling the G3 Outputs

To check the state of the G3 outputs, the read-only registers #NRrrr.17, #NRrrr.18 and #NRrrr.19 can be polled from the G3. These registers contain a copy of the actual G3 outputs. The output registers #NRrrr.14, #NRrrr.15 and #NRrrr.16 can also be polled. However, if a user sets the G3 outputs in the Kingfisher PLUS RTU and a poll occurs immediately afterwards, the user settings will be overwritten by the polled data.

Uploading Event Logs from a G3

Event logs can be uploaded from a G3 using a Kingfisher PLUS RTU and the Rx Update ladder logic block. The Kingfisher PLUS RTU will upload all the logs since the index of the last log that was uploaded. If no logs have been uploaded then the Kingfisher PLUS RTU will upload all the logs in the G3.

Appendix - RTU Data

Available data: Local Registers, Floating Point Registers, Long Registers, Network Registers, Timer Registers, IO Modules (includes PS-xx and LP-1/2/3), System Registers, Port Registers, Network Link Registers and Module Registers.

RTU Data - Local Registers

There are 2048* local registers in an RTU which can be used for general purpose data storage. Local registers can be accessed as either 16-bit registers or as single bits.

#**R**xxxx	Local register **xxxx** [1-2048]	Read/Write
#**R**xxxx.cc	Channel **cc** [1-16] of local register **xxxx** [1-2048]	Read/Write

* Additional registers are available as Network Registers. Network registers of unused RTU addresses can be used as if they were additional local registers. Providing there is enough memory allocated for Network Register Blocks, an additional 2048 network registers are available for each RTU address allowing up to 509,952 (249 x 2048) data registers per RTU.

RTU Data - Floating Point Registers

Floating Point Registers are 32-bit numbers used for storing fractional numbers, very large or very small numbers and numbers where high precision is required. Each floating point register uses two consecutive local registers and can have up to 7 digits of precision. Floating point registers can store signed numbers in the range of 3.4e-38 to 3.4e+38.

#**F**yyyy	Floating point register **yyyy** [1, 3, 5 … 2047]	Read/Write

E.g. Floating point number 100.0 is represented in hexadecimal as 0x42c80000. This would be stored as 0x0000 in odd register and 0x42c8 in the next even register, e.g.
#NA5.1 0x0000
#NA5.2 0x42c8

It is recommended that long registers be used for counting instead of floating point registers as a floating point register can only count up to 16,777,216 whereas a long register can count up to 2,147,483,647. To convert between floating point and integer format (as used by 16-bit registers), the Copy or Multi-Copy ladder blocks are used.

RTU Data - Long Registers

Long Registers are 32-bit numbers used for storing large signed integers. Each long register uses two local registers and can store numbers in the range of
-2,147,483,648 to 2,147,483,647.

#**L**yyyy	Long register **yyyy** [1, 3, 5 … 2047]	Read/Write

E.g. Long Integer number 100,500 is represented in hexadecimal as 0x18894. This would be stored as 0x8894 in odd register and 0x0001 in the next even register, e.g.
#L5.1 0x8894
#L5.2 0x0001

To convert between long and integer format (as used by 16-bit registers), the Copy or Multi-Copy ladder blocks are used.

RTU Data - Network Registers

Network Registers are used to store Hardware and Local registers from another RTU. When transferring data to another RTU, local registers are stored in the destination RTU's network registers and hardware registers are stored in the destination RTU's network analog or network digital registers. A network register can also be transferred between RTUs.

Note: network registers, network analog registers or network digital registers that have the same address as the local RTU refer to the local registers, local analog modules and local digital modules of that RTU. However, network registers that have the same address as the local RTU will not be updated if used in ladder logic.

Network Registers

Local registers received from another RTU are stored as network registers. Note: Network Float Registers and Network Long Registers are pairs of local registers received from another RTU.

#**NR**rrr.xxxx	Network register **xxxx** [1-2048] for RTU **rrr** [1-249]	Read/Write
#**NR**rrr.xxxx.cc	Channel **cc** [1-16] of network register **xxxx** [1-2048] for RTU **rrr** [1-249]	Read/Write

Network Analog Registers

Hardware analog inputs or outputs received from another RTU are stored as network analog registers.

#**NA**rrr.ss.c	Channel **c** [1-8] of analog module in slot **ss** [1-64] for RTU **rrr** [1-249]	Read/Write*

* Cannot be written if the network analog register has the same address as the local RTU.

Network Digital Register

Hardware digital inputs or outputs received from another RTU are stored as network digital registers

#**ND**rrr.ss	All 16 channels of digital module in slot **ss** [1-64] for RTU **rrr** [1-249]	Read/Write*
#**ND**rrr.ss.cc	Channel **cc** [1-16] of digital module in slot **ss** [1-64] for RTU **rrr** [1-249]	Read/Write*

* Cannot be written if the network digital register has the same address as the local RTU.

RTU Data - Timer Registers

Timer Registers are used with the Timer ladder blocks. Note: many standard periodic contacts are available as system registers (#YTICK) for which timer registers are not required. Each timer register can only be used once in ladder logic.

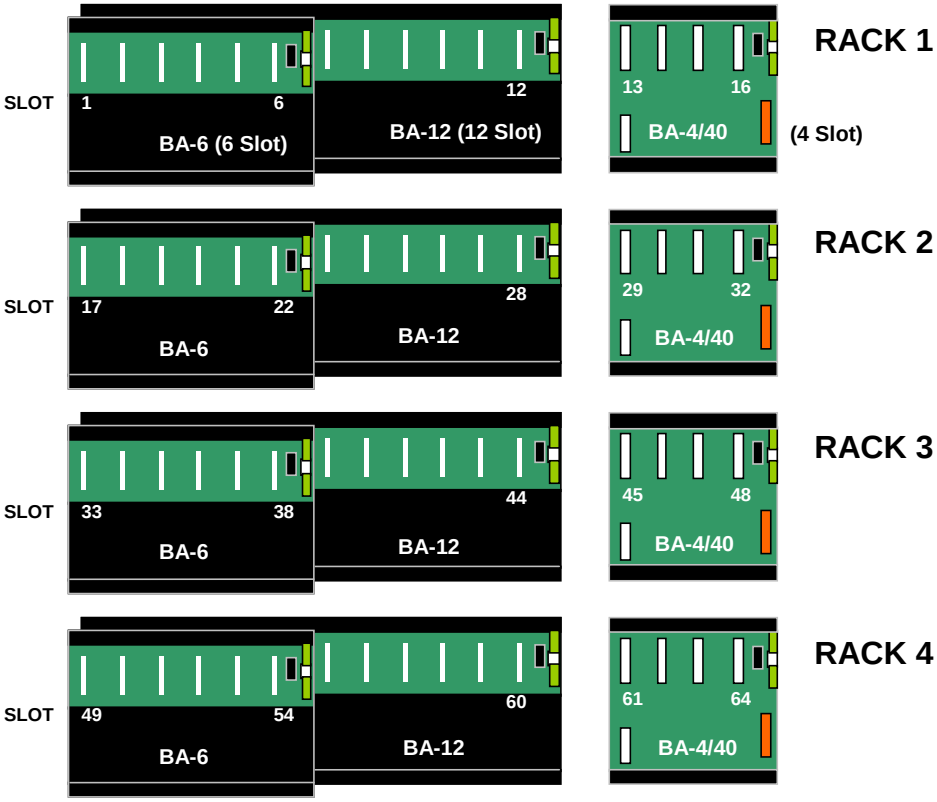
#**T**tt	Timer register **tt** [1-64]. Possible values = 0 to 65535 [16-bit]	Read/Write

RTU Data - IO Modules

Data for the following IO Modules is detailed below: AI-1/4, AI-10, AO-2/3, DI-1, DI-5, DI-10, DO-1, DO-2/5/6, IO-2, IO-3, IO-4, LP-1/2/3, PC-1, PS-10/11/12/20/21/22, PSU-1/PS-1 and RT-1.

To read data from or write data to each IO module, the module's slot address (or position on the backplane) must be known.

A Kingfisher RTU can have up to 64 modules, all installed on backplane(s). If two or more backplanes are required, they are linked together using data cables. An RTU can have up to four racks of 16 modules each. The rack number is set using switches on the backplane. A module's slot address can be determined from the type of backplane and the rack number (1-4) of the backplane as illustrated below.



If Toolbox is able to communicate with the RTU, the slot addresses of all the modules in the RTU can be viewed from the Hardware Overview (View, Hardware Overview).

Most IO modules have eight internal 16-bit registers for storing their inputs and outputs. An RTU can have four types of inputs or outputs:

- **Digital input**: an open or closed contact switched into the RTU (stored as a single bit in a register)
- **Digital output**: an open or closed relay switched out of the RTU (stored as a single bit in a register)
- **Analog input**: a 0-20mA or 4-20mA variable current (or voltage) into the RTU (stored as one input per register)
- **Analog output**: a 0-20mA or 4-20mA variable current (or voltage) out of the RTU (stored as one output per register)

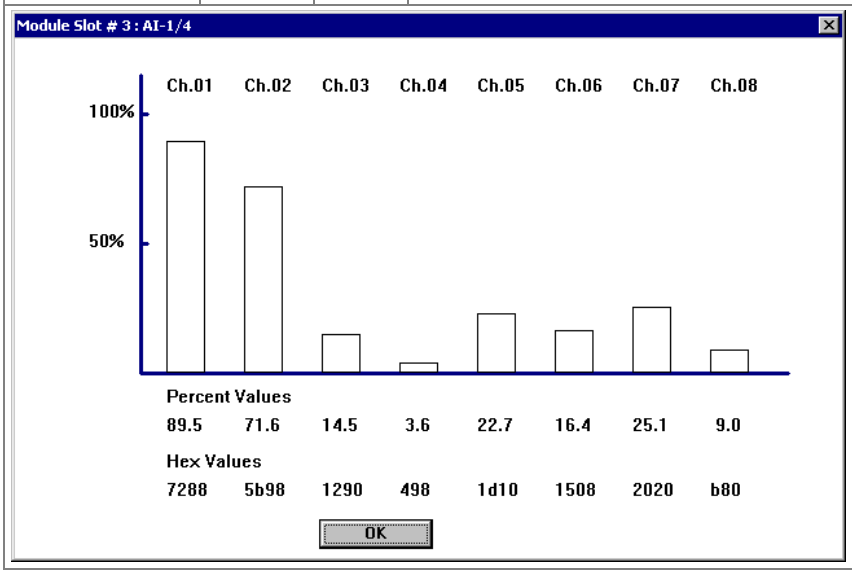
When an IO module is installed on the backplane, it is automatically detected and scanned by the RTU.

Some modules have a combination of input and output types and are called Multi-IO modules. Multi-IO modules use their 8 internal registers for both analog and digital values. The first register is used for the digital channels and the other registers (registers 2 to 7) are used for the analog channels. Therefore, to read the first analog channel from a multi IO module, the second internal register must be read (eg. the first analog input channel for an IO-3 in slot 14 is #AI14.2).

Caution! CPU and MC modules also have hardware registers that can be set using the #DO and #AO data addresses. Writing to these registers will cause the CPU or MC module to behave unpredictably and should be avoided.

AI-1/4

Register (ss = slot 1-64)	Raw Scale	Read Write	Description
#AIss.1 to 8	0-32760	Read	Analog input channels 1 to 8. Uses a 12-bit analog to digital converter. Analog values are stored as 16-bit numbers by left shifting the number by 3 bits and adding a leading sign bit (not used). Analog values are thus stored in the RTU as a number from 0 to 32760 (maximum value is 32767 minus 7 as the lower 3 bits are not used and are set to 0).

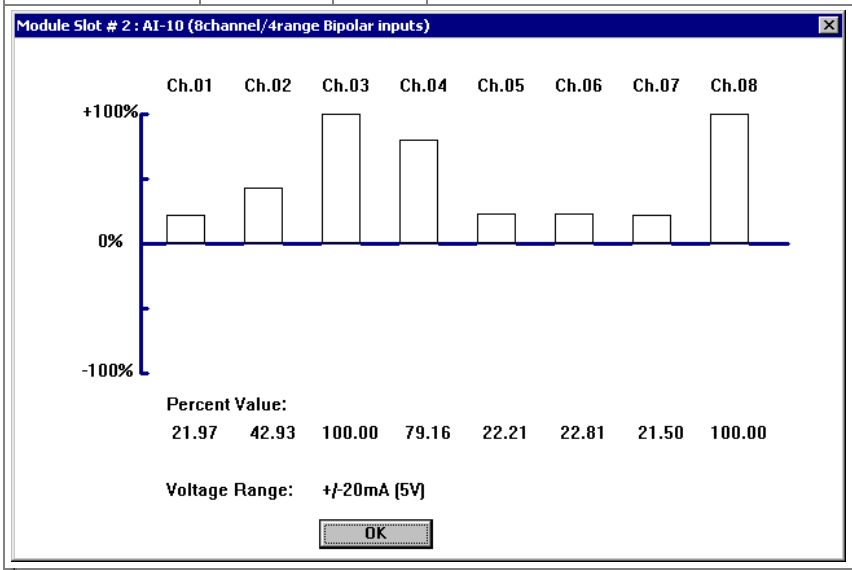


Hardware Overview

Hex Values: The raw values of the inputs (0-32760) converted to the Hexadecimal number format.

AI-10

Register (ss = slot 1-64)	**Raw Scale**	**Read/ Write**	**Description**
#AIss.1 to 8	Signed -32768 to +32767	Read	Analog input channels 1 to 8. Uses a 16-bit analog to digital converter. Analog values are stored as 15-bit numbers with a leading sign bit. The sign bit is set when the current or voltage input is negative.
#AIss.12 Bits 1 to 8 *	N/A	Read	Channels 1 to 8 under range respectively. Triggered when input is less than 4mA for the 4-20mA range or when the input is negative for the 0-20mA range.
#AIss.12 Bits 9 to 16 *	N/A	Read	Channels 1 to 8 over range respectively. Triggered when input is greater (more positive or more negative) than the configured range.

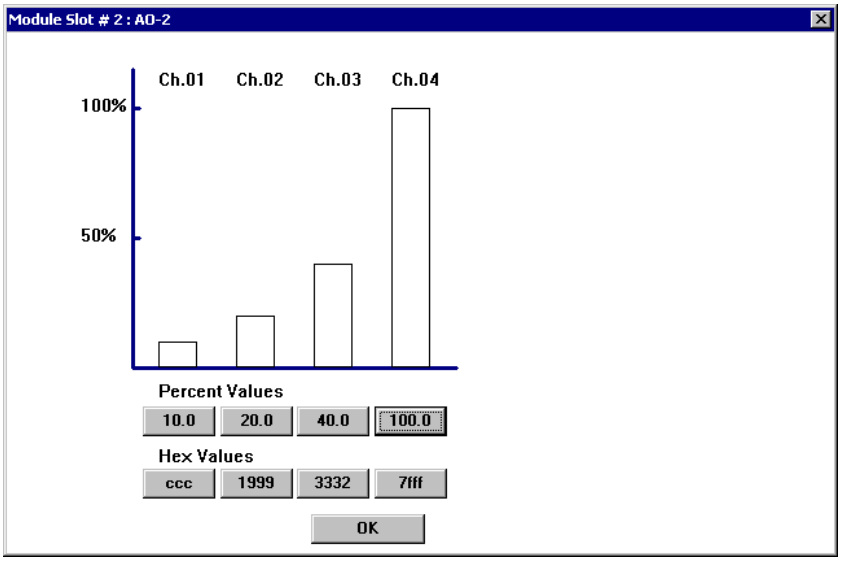


Hardware Overview

* Status bits are stored in an analog register in the module. To access these as digital bits, copy the analog register to a local register. Eg. Copy #AIss.12 to #R10 using ladder logic. #R10.1 is then channel 1 under range status, #R10.9 is channel 1 over range status etc.

AO-2/3

Register (ss = slot 1-64)	Raw Scale	Read Write	Description
#AOss.1 to 4	0-32760	R/W	Analog output channels 1 to 4. Uses a 12-bit DAC.
#AIss.5 bits 1 to 4*	N/A	Read	[AO-3 only] ON = channels 1 to 4 open loop respectively



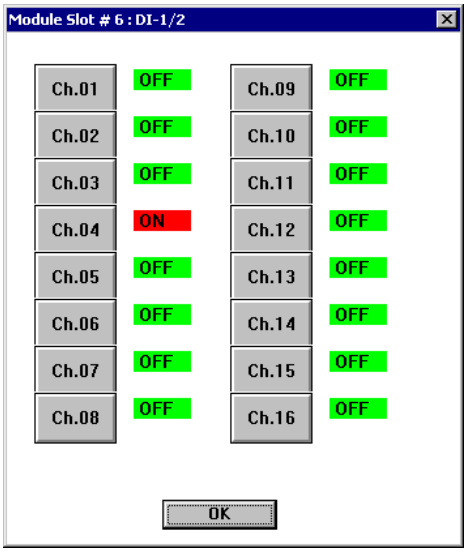
Hardware Overview

Hex Values: The raw values of the outputs (0-32760) converted to the Hexadecimal number format.

* Status bits are stored in an analog register in the module. To access these as digital bits, copy the analog register to a local register using ladder logic. Eg. For an AO-3, copy #AIss.5 to #R11, #R11.1 is then channel 1 open loop status.

DI-1

Register (ss = slot 1-64)	Raw Scale	Read/ Write	Description
#DIss.1 to 16	N/A	Read	Digital input channels 1 to 16
#DIss	N/A	Read	All 16 channels of the digital input module

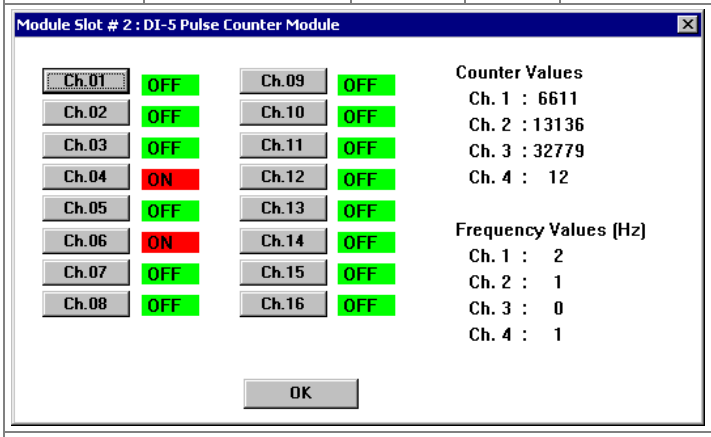


Hardware Overview

DI-5

The DI-5 counts pulses on the first four channels. Note: pulses can also be counted using any other digital input (eg. DI-5 Chs 5-16, DI-1, IO-2, IO-3, IO-4) using ladder logic. Please see the topic Example - Counting Pulses and Starts for more information.

Internal Register	Register (ss = slot 1-64)	Raw Scale	Read Write	Description
1	#DIss.1 to 16	N/A	Read	Digital input channels 1 to 16
2	#AIss.2	0-65535	R/W	Digital Input 1 Total Pulses. Write to #AOss.2 to clear.
3	#AIss.3	0-65535	R/W	Digital Input 2 Total Pulses. Write to #AOss.3 to clear.
4	#AIss.4	0-65535	R/W	Digital Input 3 Total Pulses. Write to #AOss.4 to clear.
5	#AIss.5	0-65535	R/W	Digital Input 4 Total Pulses. Write to #AOss.5 to clear.
6	#AIss.6	0-65535	R/W	Digital Input 1 Pulse Rate (Hz). Automatically cleared.
7	#AIss.7	0-65535	R/W	Digital Input 2 Pulse Rate (Hz). Automatically cleared.
8	#AIss.8	0-255	R/W	Low Byte: DI 3 Pulse Rate * (Hz). Automatically cleared. High Byte: DI 4 Pulse Rate * (Hz)



Hardware Overview

Examples	
#DI3.5	Digital module in slot 3, digital input 5 status
#AI3.2	Digital module in slot 3, digital input 1 total pulses (read this address to obtain the total)
#AO3.2	Digital module in slot 3, digital input 1 total pulses (write to this address to clear the total)
#AI3.6	Digital module in slot 3, digital input 1 pulse rate

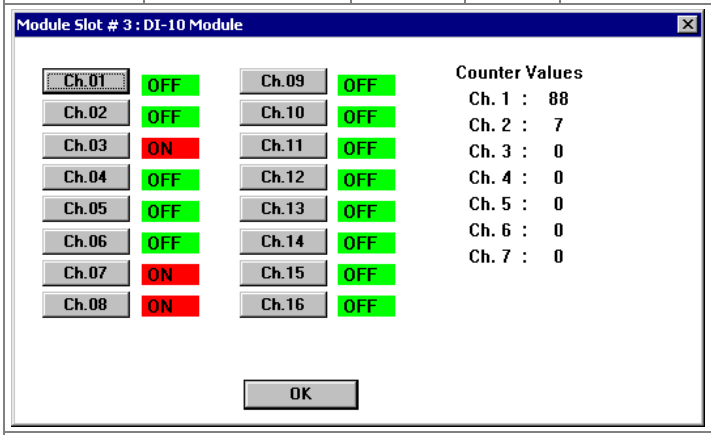
* If the maximum pulse rate is exceeded (>255 Hz), each byte will contain the lowest 8 bits of the actual pulse rate. Results are unpredictable at pulse rates greater that 1 kHz.

DI-10

The counter totals for up to 7 user-defined input channels are stored in internal registers 2 to 8 of the DI-10 and are detailed below. Note: up to 50 Hz (approx.) pulses can also be counted using a standard digital input (eg. IO-4) using ladder logic. Please see the topic Example - Counting Pulses and Starts for more information.

The DI-10 module is configured using the Toolbox menu **Configuration, IO Modules List, Configure** [DI-10]. Please see the topic Configuration - DI-10 for more information.

Internal Register	**Register** (ss = slot 1-64)	**Raw Scale**	**Read Write**	**Description**
1	#DIss.1 to 16	N/A	Read	Digital input channels 1 to 16
2 to 8	#AIss.2 to 8	0-10000 or 0-65535	Read/ Write	Channel *x* (as configured in the IO Modules list) frequency (0-10000 Hz max.) or total pulses (0-65535) or quadrature count (0-65535). Write to #AOss.2 to 8 respectively to clear pulse or quadrature totals. Frequency is automatically cleared.

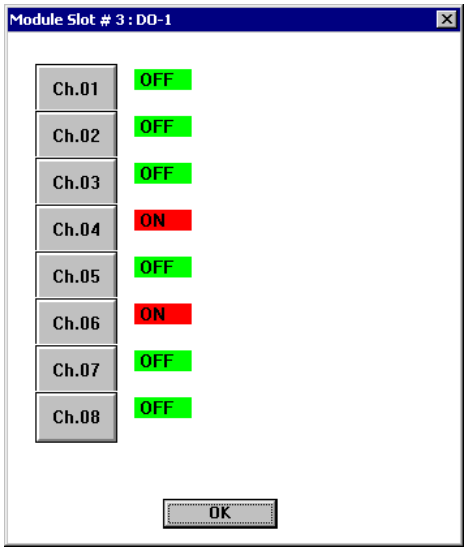


Hardware Overview

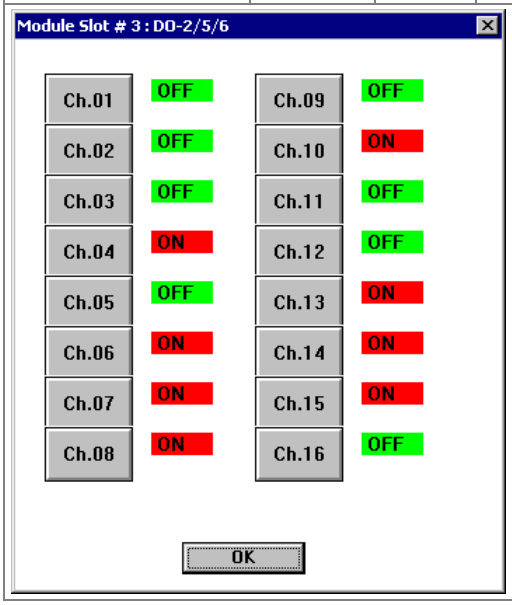
Hex Values: The raw values of the inputs or output (0-32760) converted to the Hexadecimal number format.

Examples (DI-10 has been configured to count pulses on channel 1 and count frequency on channel 2)	
#DI9.5	Digital module in slot 9, digital input 5 status
#AI9.2	Digital module in slot 9, digital input 1 total pulses (read this address to obtain the total)
#AO9.2	Digital module in slot 9, digital input 1 total pulses (write to this address to clear the total)
#AI9.3	Digital module in slot 9, digital input 2 frequency

DO-1

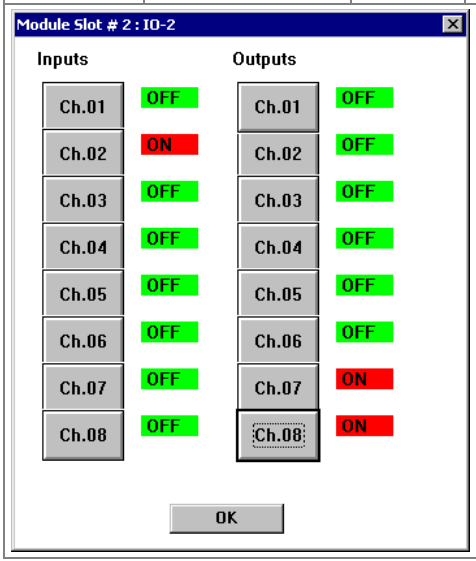
Register (ss = slot 1-64)	Raw Scale	Read Write	Description
#DOss	N/A	R/W	All 8 channels of the digital output module
#DOss.1 to 8	N/A	R/W	One of the 8 channels of the digital output module
			**Hardware Overview**

DO-2/5/6

Register (ss = slot 1-64)	Raw Scale	Read Write	Description
#DOss	N/A	R/W	All 16 channels of the digital output module
#DOss.1 to 16	N/A	R/W	One of the 16 channels of the digital output module
			**Hardware Overview**

IO-2

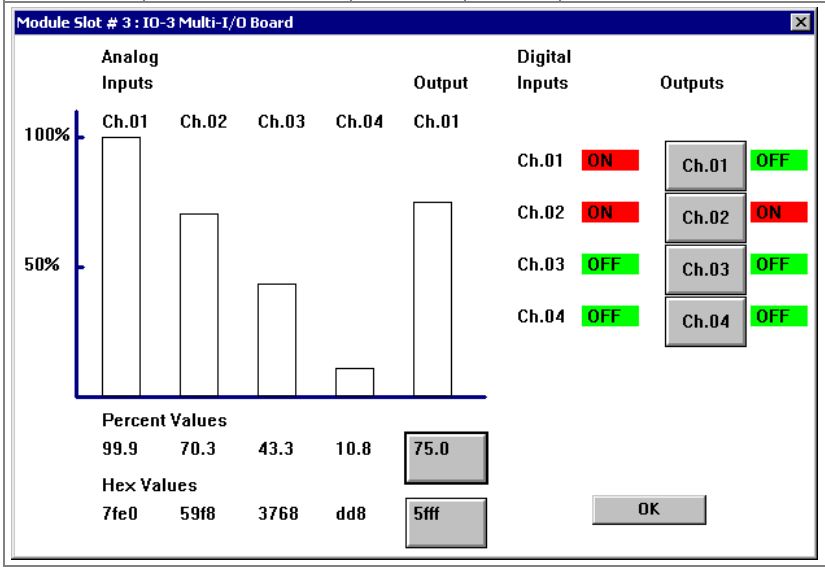
Internal Register	Register (ss = slot 1-64)	Raw Scale	Read/ Write	Description
1	#DIss.1 to 8	N/A	Read	Digital input channels 1 to 8
	#DOss.9 to 16	N/A	R/W	Digital output channels 1 to 8
2 to 8	N/A	N/A	N/A	Not Used



Hardware Overview

IO-3

Internal Register	Register (ss = slot 1-64)	Raw Scale	Read Write	Description
1	#DIss.1 to 4	N/A	Read	Digital input channels 1 to 4
	#DOss.9 to 12	N/A	R/W	Digital output channels 1 to 4
2	#AIss.2	0-32760	Read	Analog input channel 1
3	#AIss.3	0-32760	Read	Analog input channel 2
4	#AIss.4	0-32760	Read	Analog input channel 3
5	#AIss.5	0-32760	Read	Analog input channel 4
6	#AOss.6	0-32760	R/W	Analog output channel 1
7 to 8	N/A	N/A	N/A	Not used

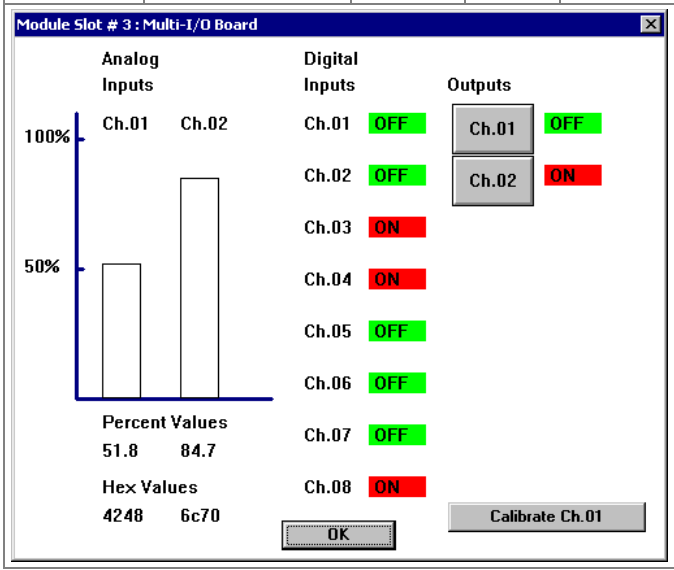


Hardware Overview

Hex Values: The raw values of the inputs or output (0-32760) converted to the Hexadecimal number format.

IO-4

Internal Register	Register (ss = slot 1-64)	Raw Scale	Read/ Write	Description
1	#DIss.1 to 8	N/A	Read	Digital input channels 1 to 8
	#DOss.9 to 10	N/A	R/W	Digital output channels 1 to 2
2	#AIss.2	0-32760	Read	Analog input channel 1
3	#AIss.3	0-32760	Read	Analog input channel 2
4 to 8	N/A	N/A	N/A	Not used

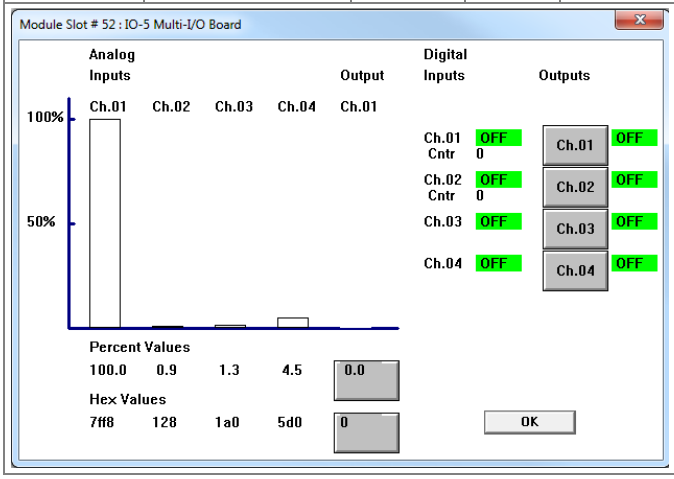


Hardware Overview

Hex Values: The raw values of the inputs (0-32760) converted to the Hexadecimal number format.

IO-5

Internal Register	Register (ss = slot 1-64)	Raw Scale	Read Write	Description
1	#DIss.1 to 4	N/A	Read	Digital input channels 1 to 4
	#DOss.9 to 12	N/A	R/W	Digital output channels 1 to 4
2	#AIss.2	0-32760	Read	Analog input channel 1
3	#AIss.3	0-32760	Read	Analog input channel 2
4	#AIss.4	0-32760	Read	Analog input channel 3
5	#AIss.5	0-32760	Read	Analog input channel 4
6	#AOss.6	0-32760	R/W	Analog output channel 1
7	#AIss.7	0-32760	Read	Digital Input 1 Counter. Use #AOss.7 to set counter value
8	#AIss.8	0-32760	Read	Digital Input 2 Counter. Use #AOss.8 to set counter value

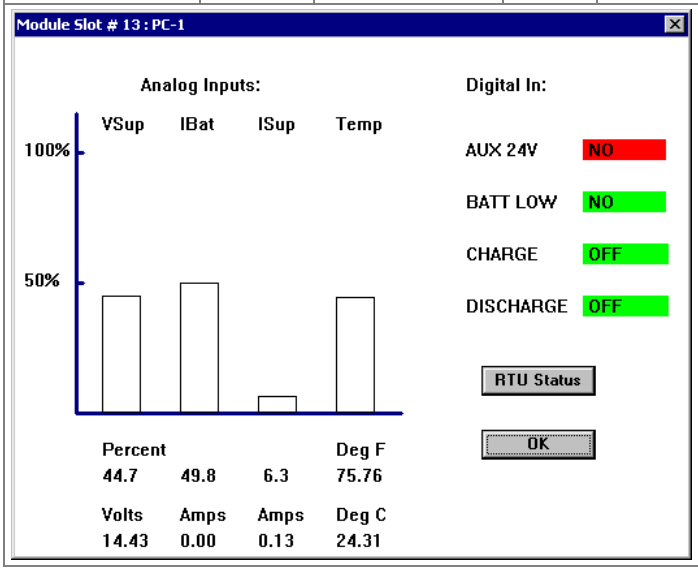


Hardware Overview

Hex Values: The raw values of the inputs or output (0-32760) converted to the Hexadecimal number format.

PC-1

Register (ss = slot 1-64)	Raw Scale	Eng. Units	Read Write	Description
#AI13.2	0-32640	0 to 32.27V	Read	PC-1 supply voltage This voltage is either the 12V supply voltage or the battery voltage. Note: the PC-1 is not AC powered.
#AI13.3	0-32640	-1 to +1A	Read	Battery charging current (positive when charging battery)
#AI13.5	0-32640	-20 to +80 °C	Read	Module temperature
#AI13.6	0-32640	0 to +2 A	Read	Supply current into the PC-1
#DI13.2	N/A	N/A	Read	ON = Aux 24V Failure or Not Present
#DI13.3	N/A	N/A	Read	ON = Battery Not Low, OFF = Battery Low
#DI13.11	N/A	N/A	Read	ON = Battery is being charged (Current into battery > 100mA)
#DI13.12	N/A	N/A	Read	ON = Battery is being discharged (Current out of battery > 60mA)
N/A	N/A	N/A	Read	AC mains fail *



Hardware Overview

* A PC-1 is powered by +12VDC and so cannot directly monitor AC mains power. To determine AC mains power fail, the battery discharging input is monitored for when it is continuously ON for more than 30 seconds. An example of how to do this is detailed in the topic Examples - Exception Reporting Digitals.

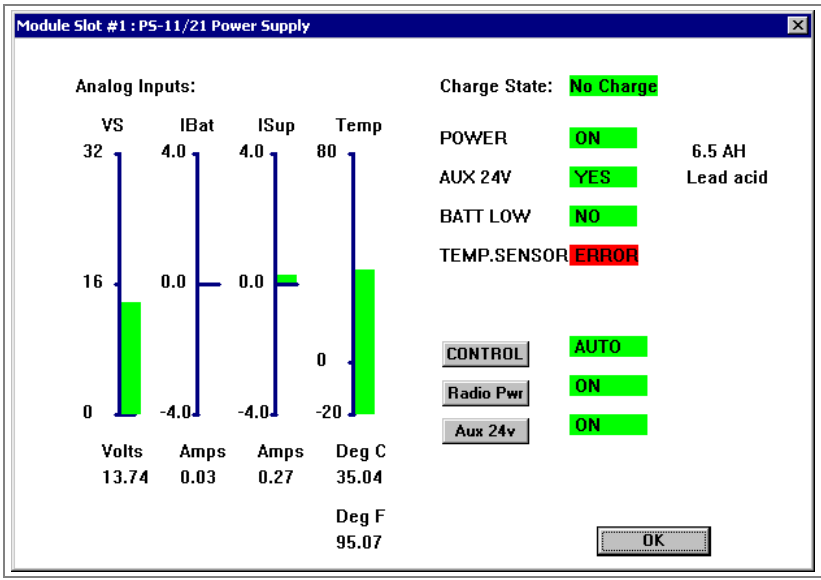
RT-1

Register (ss = slot 1-64)	Raw Scale	Read/ Write	Description
#AIss.1 to 4	0-32760 = -150 to +400 °C	Read	4-channel Pt100 temperature input module with a standard temperature range as shown. Note: other temperature ranges are available.

PS-10/11/12/20/21/22

Register (ss = slot 1-64)	**Raw Scale**	**Eng. Units**	**Read Write**	**Description**
#AIss.2	0-32736	0 to 32.27V	Read	Supply voltage - the DC voltage supplied to the RTU modules on the backplane (typically 12V) and used to charge the battery. This voltage is sourced from the battery if there is no input supply present.
#AIss.3	0-32736	-4 to +4 A	Read	Battery charge/discharge current. Current is positive when charging.
#AIss.4	0-32736	-4 to +4 A	Read	Total current supplied by the power supply to the RTU modules and battery.
#AIss.5	0-32736	-20 to +80 °C 0 °C = 6547	Read	Battery temperature (uses external temperature sensor if present, otherwise returns the same value as #AIss.14)
#AIss.7	N/A	N/A	Read	[PS-11/21 only] Configured Battery parameters Chs 1-8 = Battery Type. 0 = Default, 1 = Lead-Acid, 2 = Ni-Cad Chs 9-16 = Battery Size (x 0.1AH) 0 to 250 = 0 to 25.0AH Max.
#AIss.14	0-32736	-20 to +80 °C 0 °C = 6547	Read	Module temperature (PS-11/21 only).

#DIss.1	N/A	N/A	Read	ON = AC Power ON (PS-10/11 only) ON = DC Input Power ON (PS-20/21 only)
#DIss.2	N/A	N/A	Read	ON = Aux 24V Failure or Not Present
#DIss.3 *	N/A	N/A	Read	ON = Battery NOT Low, OFF = Battery Low
#DIss.4	N/A	N/A	Read	[PS-11/21 only] Power supply model ON = PS-21, OFF = PS-11
#DIss.5	N/A	N/A	Read	ON = Float State
#DIss.6	N/A	N/A	Read	ON = Charge State
#DIss.7	N/A	N/A	Read	ON = Boost State
#DIss.8	N/A	N/A	Read	ON = Temperature Sensor Error
#DOss.9	N/A	N/A	Read Write	ON = Manual control of radio and 24V power (and Mains Supply for PS-11) OFF = Automatic control (default)
#DOss.10	N/A	N/A	Read Write	ON = Set Radio power OFF (only when manual control enabled)
#DOss.11	N/A	N/A	Read Write	ON = Set Aux 24V OFF (only when manual control enabled)
#DOss.12	N/A	N/A	Read Write	ON = Inhibit PS-11 AC Supply Input Circuit (only when manual control enabled)



* This bit does not indicate if a battery is present as Battery Low is cleared whenever the input supply is active. If the input supply is OFF (#DIss.1=0), a battery is present if the RTU is still running!

PSU-1 / PS-1

- **PSU-1 Version 0:** 24VDC output to backplane, includes an on board fan
- **PSU-1 Version 1:** 24VDC output to backplane, same as version 0 but does not have a fan and has a different temperature measurement.
- **PS-1:** 12VDC or 24VDC output to backplane with processor controlled battery charging.

Digital Inputs
(ss = slot address 1-64)

Description	PS-1	PSU1 V0	PSU1 V1	Read / Write
ON = AC Power ON	#DIss.1	#DIss.1	#DIss.1	Read
ON = Aux 24V Failure / Not Present	#DIss.2	#DIss.2	#DIss.2	Read
ON = Battery OK (OFF = battery low)	#DIss.3	#DIss.3	#DIss.3	Read
ON = AC/DC operation (OFF=solar)	N/A	#DIss.4	#DIss.4	Read
ON = Temp > 50 deg C	N/A	N/A	#DIss.9	Read
ON = Float State	#DIss.5	N/A	N/A	Read
ON = Charge State	#DIss.6	N/A	N/A	Read
ON = Boost State	#DIss.7	N/A	N/A	Read
ON = Temperature Sensor Error	#DIss.8	N/A	N/A	Read
ON = Battery is being charged (Current into battery > 100mA)	N/A	N/A	N/A	Read
ON = Battery is being discharged (Current out of battery > 60mA)	N/A	N/A	N/A	Read

Digital Outputs
(ss = slot address 1-64)

Register	PS-1	PSU1 V0	PSU1 V1
#DOss.9	ON = Supply Voltage Trimming *	ON = Fan ON #	N/A
#DOss.10	ON = Supply Voltage Trimming *	ON = Charge control off #	ON = Charge control off #
#DOss.11	ON = Supply Voltage Trimming *	N/A	N/A
#DOss.12	ON = Supply Voltage Trimming *	N/A	N/A
#DOss.13	ON = Supply Voltage Trimming *	N/A	N/A
#DOss.14	N/A	N/A	N/A
#DOss.15	N/A	N/A	N/A
#DOss.16	ON = Manual Trim Control	N/A	N/A

* Supply Voltage Trimming (0-31). The number of voltage steps between the minimum and maximum DC output voltage. For a 12V PS-1, there are 31 steps of approximately 100mV between 12 and 15 volts. For a 24V PS-1 there are 31 steps of approximately 250mV between 24 and 32 Volts.
# Only controllable during first 5 minutes after switching on.

Analog Inputs
(ss = slot address 1-64)

Module	Register (ss = slot 1-64)	Raw Scale	Eng. Units	Read Write	Description
PSU1 V0	#AIss.2:	0-32640	0 to 32.27V	Read	Supply voltage to power supply
	#AIss.3:	0-32640	-2 to +2 A	Read	Battery charging current
	#AIss.4:	0-32640	0 to 32.27 V	Read	Battery Voltage
	#AIss.5:	16128-32640	-25 to +227° C	Read	Module temperature
PSU1 V1	#AIss.2:	0-32640	0 to 32.27V	Read	Supply voltage to power supply
	#AIss.3:	0-32640	-2 to +2 A	Read	Battery charging current
	#AIss.4:	0-32640	0 to 32.27 V	Read	Battery Voltage
	#AIss.5:	Signed; -18688 to 32576 (0°C=0)	-55 to +125° C	Read	Module temperature
PS-1	#AIss.2:	0-32736	0 to 32.27V	Read	Supply voltage - the DC voltage supplied to the RTU modules on the backplane (typically 12V) and used to charge the battery. This voltage is sourced from the battery if there is no input supply present.
	#AIss.3:	0-32736	-4 to +4 A	Read	Battery charge/discharge current. Current is positive when charging.
	#AIss.4:	0-32736	-4 to +4 A	Read	Total current supplied by the power supply to the RTU modules and battery.
	#AIss.5:	0-32736 (0° C=6528)	-20 to +80°C	Read	Module temperature

LP-1/2/3

Note: Only read/write hardware registers listed here should be written to. Other hardware registers are used by the LP-1/2/3 firmware and care must be taken not to overwrite these.

LP-1/2/3 I/O Registers

Register	Description	Comments	Read/Write
#DI1.1	Digital input 1	0V or Open =OFF. Pulse input 1	Read
#DI1.2	Digital input 2	0V or Open =OFF. Pulse input 2	Read
#DI1.3	Digital input 3	0V or Open =OFF.	Read
#DI1.4	Digital input 4	0V or Open =OFF.	Read
#DI1.5	Digital input 5	0V or Open =OFF. Can be used as output #DO2.3	Read
#DI1.6	Digital input 6	0V or Open =OFF. Can be used as output #DO2.4	Read
#DI1.7	Digital input 7	0V or Open =OFF. Can be used as output #DO2.5	Read
#DI1.8	Digital input 8	0V or Open =OFF. Can be used as output #DO2.6	Read
#DO2.1	Digital output 1	Latched relay output (1 = closed)	Read/Write
#DO2.2	Digital output 2	Latched relay output (1 = closed)	Read/Write
#DO2.3	Digital output 3	Optional Transistor output (1 = closed ie. sink to 0V)	Read/Write
#DO2.4	Digital output 4	Optional Transistor output (1 = closed ie. sink to 0V)	Read/Write
#DO2.5	Digital output 5	Optional Transistor output (1 = closed ie. sink to 0V)	Read/Write
#DO2.6	Digital output 6	Optional Transistor output (1 = closed ie. sink to 0V)	Read/Write
#AI1.2	Analog input 1	12bit, 0-32760 = 0-100%	Read
#AI1.3	Analog input 2	12bit, 0-32760 = 0-100%	Read
#AI1.4	Analog input 3	12bit, 0-32760 = 0-100%	Read
#AI1.5	Analog input 4	12bit, 0-32760 = 0-100%	Read
#AI2.4	Pulse counter 1	0-65535 pulses. Write 0 to #AO2.4 to reset	Read
#AI2.5	Pulse counter 2	0-65535 pulses. Write 0 to #AO2.5 to reset	Read
AO5.4	Analog Output 1	15 bit, 0-32767 = 0-100%. LP-3 Only.	Read/Write

LP-1/2/3 Power Supply Data

Register	Description	Comments	Read/Write
#AI1.7	Ambient temperature	0-32760 = -23 to 77 °C* (LP-1) 0-32760 = -60 to 100 °C* (LP-2/3) Accuracy: ± 3%	Read
#AI1.8	DC INPUT supply voltage to LP-1/2/3 **Mains Fail**: If the supply voltage drops dramatically, it can be assumed that the LP-1/2/3 is now running off the battery and mains fail has occurred. e.g. If #AI1.8>17690 (13.5V) = Mains OK If #AI1.8<17035 (13.0V) = Mains Fail	0-32760 = 0 to 25V Accuracy: ± 2%	Read
#AO2.9 Chs 1-8	Low Voltage Wakeup Conditions. The initial setting in this register specifies which wakeup triggers to enable. When the RTU wakes up (after the supply voltage is restored), the wakeup trigger that was activated is written to this register by the RTU. Ch1 Time wakeup (automatically set) Ch 2 Digital input 1 Change Of State Ch 3 Digital input 2 Change Of State Ch 4 Port 4 hardware wakeup (LP-1 only) Ch 5 Port 1 CTS change Ch 6 Port 2 CTS change (LP-1 only) Ch 7-8 Not used	**LP-1 default:** #AO2.9 = 0. When #AO2.9 Chs 1-8 are set to 0, the RTU will use 18 Hex (chs 4 & 5 =1) for the Low Voltage Wakeup Conditions. **LP-2/3 default:** #AO2.9 = 10 Hex (Ch 5 = 1). When #AO2.9 Chs 1-8 are set to 0, the RTU will use 10 Hex (Ch 5 = 1) for the Low Voltage Wakeup Conditions.	Read/Write
#AO2.9 Chs 9-16	Low Voltage Monitoring Interval 0 = Do not monitor supply voltage (LP-2/3) or monitor supply voltage every 5 minutes (LP-1) 1 to 254 = check supply volts every 1 to 254 minutes respectively 255 = disable low voltage shutdown (LP-1)	Default = 0 LP-2/3: Set #AO2.9 = 256 or 16#100 (100 Hex) or similar in ladder logic to enable low voltage shutdown and monitoring of the supply every minute. Note: **5V Out** must be enabled to allow the LP-1/2/3 to monitor the supply voltage (please see #DO2.12 below)	Read/Write
#AO2.10	Input Supply Shutdown/Startup voltages Chs 1-8 (1-255 x 0.1 V) = LP-1/2/3 shutdown voltage Chs 9-16 (1-255 x 0.1 V) = LP-1/2/3 startup voltage	Default: AO2.10=7668 Hex Chs 1-8 = 104 (68H) = 10.4V shutdown Chs 9-16 = 118 (76H) = 11.8V startup	Read/Write

* The LP-1 can only measure temperatures 0 °C and above. The negative temperature limit allows for future development.

LP-1/2/3 Hardware Setup Registers

Register	Description	Default	Comments	Read/Write
#DI1.9	LP-1: Port 3 Status	0	LP-1: 1 = port 3 configured LP-2/3: Not used.	Read
#DI1.10	LP-1: Port 4 Status LP-2/3: Port 3 or Port 4 detected	0	LP-1: 1 = port 4 configured LP-2/3: 1 = one or two option ports detected	Read
#DO1.13	Pulse input 1 enable	0 (LP-1) 1 (LP-2/3)	1 = enabled. Also allows wakeup via DI1	Read/Write
#DO1.14	Pulse input 2 enable	0 (LP-1) 1 (LP-2/3)	1 = enabled. Also allows wakeup via DI2	Read/Write
#DO1.15	Pulse input 1 active transition	0	0 = low to high transition 1 = high to low	Read/Write
#DO1.16	Pulse input 2 active transition	0	0 = low to high transition 1 = high to low	Read/Write
#DO2.7	DO3 enable	0	0 = DO3 disabled & DI5 enabled	Read/Write
#DO2.8	DO4 enable	0	0 = DO4 disabled & DI6 enabled	Read/Write
#DO2.9	DO5 enable	0	0 = DO5 disabled & DI7 enabled	Read/Write
#DO2.10	DO6 enable	0	0 = DO6 disabled & DI8 enabled	Read/Write
DO2.11	LP-1: **12 V Out** enable LP-2/3: **VDC1 Out** enable	0 (LP-1) 1 (LP-2/3)	LP-1: 0 = **12V Out** Disabled. **12V Out** is used to power a port 3 Trio radio if installed. LP-2/3: 1 = **VDC1 Out** enabled	Read/Write
DO2.12	I/O, **5V Out** (LP-1/2/3) and **12V Aux** (LP-1) disable	0	0 = I/O, **5V Out** and **12V Aux** (LP-1) enabled. When set (=1): • Switches off **5V Out** and **12V Aux** (LP-1) • Stops updating analog and digital inputs • Digital inputs 1 & 2 still operate • Digital outputs 1 & 2 still operate • Disables power to ports 3 and 4 (LP-2/3) Note 1: I/O and **5V Out** must be enabled for low voltage shutdown to operate correctly (allows reading of the supply voltage) Note 2: **5V Out** is used to power a port 3 internal GSM if installed (LP-1). Note 3: **CPU 5V Out** remains enabled	Read/Write
DO2.13	LP-1: Port 3 disable LP-2/3: **VDC2 Out** enable	1	LP-1: 1 = Port 3 disabled LP-2/3: 1 =**VDC2 Out** enabled	Read/Write
DO2.14	LP-1: Port 4 disable LP-2/3: Port 3 and Port 4 disable	1 (LP-1) 0 (LP-2/3)	LP-1: 1 = Port 4 disabled LP-2/3: 0 = Port 3 and Port 4 enabled	Read/Write
#AO5.5	Pulse Output count	0	Uses DO Ch3 (must be enabled)	Read/Write
#AO5.6	Pulse ON time	0	x 1ms	Read/Write
#AO5.7	Pulse OFF time	0	x 1ms	Read/Write
#DO6.1	Analog input 1 live zero	0	0=0-5 V, 1=1-5 V	Read/Write

#DO6.2	Analog input 2 live zero	0	0=0-5 V, 1=1-5 V	Read/Write
#DO6.3	Analog input 3 live zero	0	0=0-5 V, 1=1-5 V	Read/Write
#DO6.4	Analog input 4 live zero	0	0=0-5 V, 1=1-5 V	Read/Write
#AO3.9, #AO3.10	Pulse 1 (DI1) or pulse 2 (DI2) debounce time respectively	0	0 to 32000 ms debounce time The pulse must remain in the active state for the debounce time before a pulse is counted (0 = no debounce).	Read/Write
#AO4.2, #AO4.3, #AO4.4, #AO4.5	Analog input 1 to 4 average sample count respectively	1	1-59 samples. Number of samples to average the analog input over before updating the analog input data register. Each sample takes up to 110 ms.	Read/Write

LP-1/2/3 Miscellaneous Data

Register	Description	Comments	Read/Write
#AI2.6	Trio SR radio RSSI (LP-1 only)	0-32760 = 0-5 V 2V = -90 dBm, 10 dB per 0.5 V steps	Read
#AI2.7	LP-1: Trio SR radio TX power LP-2/3: Port 3 option board type detected	LP-1: 0-32760 = 0-5V = Vt (volts). Tx Power (W) = 1.151 x $(Vt + 0.192)^2$ Note: the TX power of the Trio radio must be programmed to 2 watts or less (1 watt is recommended) to prevent overloading of the LP-1. Trio radios with higher TX power settings (eg 5 watts) will cause a power dip in the LP-1 when transmitting - resulting in comms failure. LP-2/3: 0=none, 1=serial/fibre optic, 2=pstn, 3=pline, 4=Ethernet, 5=image, 6=image (running)	Read
#AI2.8	Port 4 option board type detected	0=none, 1=serial/fibre optic, 2=pstn, 3=pline, 4=Ethernet, 5=image, 6=image (running)	Read
#AO2.13	Ports 1 and 2 Data Format	Ch 1: Port 1 stop bits, 0=one, 1=two Ch 2: Port 1 parity, 0=even, 1=odd Ch 3: Port 1 parity enable, 0=disable, 1=enable Ch 4: Port 1 data bits, 0=eight, 1=seven Ch 5-8: Port 1 new data format enable. Must write D hex (1101 binary) to enable. Ch 9: Port 2 stop bits, 0=one, 1=two Ch 10: Port 2 parity, 0=even, 1=odd Ch 11: Port 2 parity enable, 0=disable, 1=enable Ch 12: Port 2 data bits, 0=eight, 1=seven Ch 13-16: Port 2 new data format enable. Must write D hex (1101 binary) to enable.	Read/Write
#AO2.14	Ports 3 and 4 Data Format	Ch 1: Port 3 stop bits, 0=one, 1=two Ch 2: Port 3 parity, 0=even, 1=odd Ch 3: Port 3 parity enable, 0=disable, 1=enable Ch 4: Port 3 data bits, 0=eight, 1=seven Ch 5-8: Port 3 new data format enable. Must write D hex (1101 binary) to enable. Ch 9: Port 4 stop bits, 0=one, 1=two Ch 10: Port 4 parity, 0=even, 1=odd Ch 11: Port 4 parity enable, 0=disable, 1=enable Ch 12: Port 4 data bits, 0=eight, 1=seven Ch 13-16: Port 4 new data format enable. Must write D hex (1101 binary) to enable.	Read/Write
#AI3.5	Vref	ADC reference voltage. 0-32767 = 0-5V Normally 2.5V $\pm$2%	Read
#DI3.1	Firmware error	1 = Error	Read
#DI3.2	SRAM error	1 = Error	Read

RTU Data - System Registers

System registers contain configuration parameters, general settings, real-time clock settings and various other RTU parameters. Most of these registers are read / write which means they can be changed by ladder while the RTU is running.

For the following registers, *cc* is the register channel (1 to 16) where applicable.

#YADDRESS	RTU Address (0-249)	Read/Write
#YPRIORITY	Comms Priority (0, 1 or 2)	Read/Write
#YQUIET	Quiet time before Tx (ms)	Read/Write
#YRETRIES	Global retries	Read/Write
#YTIMEOUT	Global timeout (ms)	Read/Write
#YSYSID	System ID (0-255)	Read/Write
#YIPADDR1	(CP-21 only, 0-65535) Higher half of IP address *A.B.C.D* of CP-21 port 2. Chs 1-8=A, Chs 9-16=B. To set A and B, use #YPADDR1=A+256xB. Not used for CP-10/11.	Read/Write
#YIPADDR2	(CP-21 only, 0-65535) Lower half of IP address *A.B.C.D* of CP-21 port 2. Chs 1-8=C, Chs 9-16=D. To set C and D, use #YPADDR2=C+256xD. Not used for CP-10/11.	Read/Write
#Y2NDRTU	Primary / Secondary RTU address. This parameter is used in the ladder of the secondary RTU to specify the address of the primary RTU and it is used in the ladder of the primary RTU to specify the address of the secondary RTU. Please see the appendix topic Redundancy, Redundant RTUs.	Read/Write
#Y2NDSTAT	Primary / Secondary master RTU Status. This is an integer value that can be configured as follows: **0 = No primary/secondary.** The RTU is not a primary or a secondary RTU. This is the default setting after a cold start. **1 = Primary control.** The RTU is the primary RTU and is in control. This means the RTU will behave like a normally configured RTU and will initiate and acknowledge messages. When the primary RTU is in this mode, the secondary RTU should be in secondary listen mode (mode 4 or 12). **2 = Primary listen.** This RTU is the primary RTU but it is in listen mode. This means the RTU will only respond to messages from the PC and messages from the secondary RTU. The primary RTU should start up in this mode to check if the secondary is in control mode. If the primary starts up in control mode and the secondary is also in control mode, both RTU's will acknowledge the same messages - causing communication failures. When the secondary is in control mode it should regularly test the primary to see if the primary is ready to resume control. If the primary is ready, the secondary should go back to listen mode and the primary should take over. While in this mode, the RTU will not relay Modbus write commands to the outstations (please see mode 10). **3 = Secondary control.** The secondary RTU is in control mode and will respond to messages for itself and to messages for the primary RTU. The secondary RTU should enter this mode when the primary RTU fails. When in secondary control mode, the RTU will accept all outputs to the Primary RTU's local registers and write these to its own local registers. **4 = Secondary listen.** The secondary RTU is in listen mode and will respond to messages for itself only. If it receives a message for the primary RTU it will process it (for example update its network registers) but it will not acknowledge the message. The secondary listen RTU will allow messages (eg. from Toolbox) to be relayed to the Primary RTU and to the outstation RTUs. While in this mode, the RTU will not relay Modbus write commands to the outstations (see mode 12). The secondary RTU should be in this mode when the primary RTU is in primary control mode. Note: the secondary RTU will not update its event logs when it hears event log messages for the primary RTU. Event logs must be updated in the secondary RTU by using a TX Update Event Logs block in the primary RTU. **10 = Primary Listen with relay Modbus write commands.** This has the same function as the Primary Listen mode but will also allow Modbus outputs to be relayed to the outstation RTUs. **12 = Secondary Listen with relay Modbus write commands.** This has the same function as the Secondary Listen mode but will also allow Modbus outputs to be relayed to outstation RTUs.	Read/Write

#YDIAG.cc	RTU diagnostic register		
	Ch1	Same as #YSYS.WARMST below	Read/Write
	Ch2	1 = No IO module detected on the backplane OR an IO module that was detected on the backplane is no longer present. Eg. Flag is triggered when an IO module is removed while the RTU is running. Flag is reset when the IO module is re-installed in the same slot OR after a warm start if the RTU is able to detect at least one IO module on the backplane.	Read
	Ch3	1 = No MC module detected on the backplane OR an MC module that was detected on the backplane is no longer present. Eg. Flag is triggered when an MC module is removed while the RTU is running. Flag is reset when the MC module is re-installed in the same slot OR after a warm start if the RTU is able to detect at least one MC module on the backplane.	Read
	Ch4	1 = Real time clock fail. 0 = OK. Loopback testing is only performed after a warm start.	Read
	Ch5	1 = Port 1 internal loopback fail, 0 = OK	Read
	Ch6	1 = Port 2 internal loopback fail, 0 = OK	Read
	Ch7	1 = CP-xx Port 3 internal loopback fail, 0 = OK	Read
	Ch8	Same as #YSYS.SCAN1 below	Read
	Ch9-14	1 = RAM chip 1 to 6 fail respectively, 0 = OK / RAM Chip not installed. RAM chip 1 to 6 testing is performed after a cold start. A CP-10/11 has four RAM chips and a CP-21 has six RAM chips.	Read
	Ch15	1 = Watchdog timer fail, 0 = OK. (The RTU has restarted itself since the last warm or cold start from Toolbox. Reset by a cold or warm start from Toolbox).	Read
	Ch16	1 = Network Data overflow, 0 = OK. It is possible for the CPU to run out of memory when storing a lot of network data.	Read
#YDIAG2.cc	RTU diagnostic register 2 (read only)		
	Ch1	Same as #YSYS.ENABLE below	Read/Write
#YEXCEP	CP-x Redundancy changeover status. Returns an integer number in the low byte (channels 1-8) representing the changeover status and an integer number in the high byte (channels 9-16) representing the reason for the changeover. An RTU can force a changeover itself by setting #YEXCEP=16#800. #YEXCEP will then return 900 Hex after the changeover. Please see the appendix Redundancy - Redundant CPUs.		Read/Write
	Ch1-8	Changeover status (0-255) 0 = Changeover OK 1 = Timeout 2 = CRC Error 3 to 255 = Not used	Read
	Ch9-16	Changeover reason (0-255) 0 = No changeover 1 = User changeover (using Toolbox) 2 = Redundant Duty IO bus failure 3 = Redundant Duty communications bus failure 5 = No communications on IO bus and communications bus 6 = Redundant Standby IO bus failure 7 = Redundant Standby communications bus failure 8 = Ladder changeover request (set using ladder logic) 9 = Ladder changeover performed 10 to 25 = Inactivity on port 1 to 16 respectively 26 = Logic disabled 27 = Other redundant CPU in duty mode 28 to 255 = Not used	Read/Write
#YFIRMW	Firmware version. Returns a 2-byte value corresponding to the firmware version. Eg. E139 Hex = Version 1.39E		Read

#**YFLAGS.**cc	General RTU operation flags		
	Ch1	1 = Scan overrun. Scan overrun is a status indication only and does not affect RTU operation. Triggered when the RTU cannot read all the IO modules and process the ladder logic at the configured rate (Configuration, System Parameters - IO Scan Interval).	Read
	Ch2	1 = Driver error. There was a CRC error in the firmware driver. This bit is set or reset after a warm start.	Read
	Ch3	1 = Currently synchronizing the clock.	Read
	Ch4	1 = The maximum number of TX_IMAGES are active at this moment. Updating network register blocks in the destination RTU requires a considerable amount of network management. When the network management registers are fully used, this bit is set. This would only occur when trying to run multiple TX_IMAGES blocks simultaneously.	Read
	Ch5	1 = Ladder error. CRC failed.	Read
	Ch6	1 = Log overrun. Log Overrun is a status indication only and does not affect RTU operation. Triggered when the memory allocated for event logs is full and the oldest event logs are being overwritten.	Read
	Ch7	1 = Message buffer full. The message buffer is full and no new messages can be initiated or received. The message buffer can contain a maximum of 32 messages. Occurs when messages are continuously generated faster that they can be transmitted OR after sending two or more pager messages for CPU firmware prior to 1.43d. Flag is cleared when RTU is warm started.	Read
	Ch8	1 = Pack event logs function active	Read
	Ch9	1 = Network data overflow (same as #YDIAG.16)	Read
	Ch10	1 = Tx DNP3 active	Read
#**YIMPL.**cc	Firmware Implementation. Firmware is available with a number of communication options.		
	Ch1	1 = Form 4C Recloser	Read
	Ch2	1 = ADS Data Logger	Read
	Ch3	1 = TMR Radio	Read
	Ch4	Not Used	Read
	Ch5	1 = Redundancy	Read
	Ch6	1 = MAC-800 RTU	Read
	Ch7	1 = Microtran	Read
	Ch8	1 = Minitran	Read
	Ch9	1 = Inline Flow Computer	Read
	Ch10	1 = Email Modem Switch Unit	Read
	Ch11	1 = Paging	Read
	Ch12	1 = Gas and Fuel Minitran RTU (Minitran 1)	Read
	Ch13	1 = Omron PLC	Read
	Ch14	1 = Gas Calculation AGA8	Read
	Ch15	1 = Hohner Shaft Encoder	Read
	Ch16	1 = Steam flow calculation AGA9	Read
#**YLOGIDX**	Event log current index pointer (0-65535). Each new event log is stored at the location pointed to by this register and then the register is incremented to the next event log location. Note: the event log list is a circular buffer. When the buffer is full, the oldest event logs will be overwritten.		Read/Write
#**YMATHSTAT**	Maths error status register. 0 = Maths OK.		Read

#YMODE.cc	Operation flags.		
	Ch1	1 = Qset messages enabled (default). Qset allows the RTU to manage and relay *set output messages* to an outstation RTU. These messages are usually initiated by SCADA software.	Read/Write
	Ch2	1 = Disable automatic update of network list. Default = enabled (network list is updated according to the network details of the last message received). Eg. if the network link to RTU1 is configured on port 3 and a message is received from RTU1 on port 4, then the network list is updated to show RTU1 connected via port 4.	Read/Write
	Ch3	1 = Uploads event logs only if time and date of log is newer than the last log uploaded (using RXUPD04 and newer driver). Older logs are ignored. Default = ignore time and date, upload logs from pointer onwards.	Read/Write
	Ch4	1= Keep Ethernet port open on the redundant standby CPU. The duty and standby CPUs can both have their Ethernet port working simultaneously as long as they are configured with different Ethernet addresses (allows support for redundant LANs).	Read/Write
	Ch5	1 = Disable port 2 of the redundant standby processor	Read/Write
	Ch6	1 = Disable port 3 of the redundant standby processor	Read/Write
#YPAGERS.cc	Active pager status register. Channels 1 to 12 of this register indicate which of the 1 to 12 pager receivers respectively are currently being sent a message.		Read
#YPDTIME	Power Down Time in seconds (0-65535 sec = 0-18.2 hours). Writing a number to this register causes the RTU to go into power-down mode for the specified number of seconds. While in power-down mode, the RTU switches off the various output supply voltages and puts the processor to sleep (ladder logic and IO are not scanned). This reduces the power consumption of the CPU module by more than half. When the RTU is woken up (see #YPDSTAT), the Power Down Time register is reset to zero. When in power-down mode, the *WD* and *T1* LEDs on a PC-1 continually flash and the RTU will not respond to any messages until it is woken up. The voltages that can be controlled are the same as for the IO Power Saving Control as configured in **Configuration, System Parameters** for the RTU. These voltages are: • 24V output voltage from the IO modules (eg. AI-1, AI-10, IO-3, IO-4) • 24V auxiliary supply from a PSU-1, PS-1, PS-xx or from a PC-1 • 12V Vr from a PC-1, PS-1 or PS-xx. (Note: the 12V output supply from an LM-2 or from a PC-1/MC-1 radio board is not switched off).		Read/Write
#YPDSTAT	Power down status. The power down status is used to program which port(s) can wake the RTU up and is also used to indicate how the RTU was last woken up. Writing a *1* to the appropriate channel enables that port to wake the RTU up. When the RTU is woken up, the RTU will clear the register and then set the channel corresponding to the wakeup event. An RTU is woken up after the power down time has expired (see #YPDTIME) or a *CTS* is detected on a configured serial port or a wakeup message is received from a configured radio or private line port (a wakeup message is configured by ticking the check box in the network link of the initiating RTU). A CTS is detected on a serial port when a cable with an RTS/CTS loop is connected (these are standard in ADP-05 Toolbox cables). The 16 channels of #YPDSTAT are defined as follows:		Integer Read/Write
	Ch1	Time wake-up	Read
	Ch2-16	Ports 1-15 wake-up respectively	Read/Write

#YPDSTAT (LP-2/3)	#YPDSTAT is defined differently for LP-x RTUs than for other types of Kingfisher RTUs as follows: Configuration: #YPDSTAT Wakeup Conditions (integer read/write) Ch 1 Time wakeup (default setting. Always set when #YPDTIME is set) Ch 2 Digital input 1 change of state Ch 3 Digital input 2 change of state Ch 4 Not Used Ch 5 Port 1 CTS change (default setting) Chs 6-15 Not Used Ch 16 Sub-active sleep mode. The LP-2/3 remains in sleep mode while counting pulses from digital inputs 1 or 2. e.g. Can enter sub-active sleep mode by writing 16#8010 (8010 Hex) to #YPDSTAT (this will allow the LP-2/3 to count pulses and will also enable wakeup on port 1 CTS change). 5V Out, V1 Out and 24V Out are also disabled in this mode	
#YRELAYRTU	The RTU address of the last Series I RTU that a message was relayed to. This parameter can be used by a master RTU that relays output messages from a computer.	Read
#YRTUTYPE	RTU processor type. 1 = CP-1, 2 = PC-1, 3 = CP-10/11, 4 = CP-21, 5 = SBX, 6 = ERS Micro, 7 = LP-1/2/3, 8 = SB-1, 9 = Micro-4	Read
#YSOCKETx	(CP-21 only) Returns the network usage of the 24 possible CP-21 Ethernet sockets (where x=socket number) on the LAN. 0=not used, 1-255=the address of the RTU using that socket.	Read

#YSTAT.cc	RTU status register		
	Ch1	Please see Chs 6-8	Read
	Ch2	1 = CPU in slave mode, 0 = CPU in master mode	Read
	Ch3	1 = I/O scanning enabled, 0 = IO scanning disabled	Read
	Ch4	1 = logic enabled, 0 = logic disabled	Read
	Ch5	1 = RTU in program mode (IO and logic scanning disabled) 0 = RTU in run mode	Read
	Ch1, Ch6-8	CPU / RTU type	Read

Ch8	Ch7	Ch6	Ch1	Description
0	0	0	0	CP-1
0	0	0	1	SBX-2
0	0	1	0	PC-1
0	0	1	1	PC-1
0	1	0	0	CP-20/21
0	1	1	0	CP-10/11
1	0	1	0	ERS
1	0	1	1	Micro-3
1	1	0	0	LP-1/2/3
1	1	0	1	Micro-4

	Ch12-14	RAM size	Read
		LP-1/2/3, PC-1, CP-1, SBX-2	CP-10/11, CP-20/21
	Ch12	1 = RAM >= 128 K 0 = RAM < 128 K	1 = RAM >= 512 K 0 = RAM < 512 K
	Ch13	1 = RAM >= 256 K 0 = RAM < 256 K	1 = RAM >= 1024 K 0 = RAM < 1024 K
	Ch14	1 = RAM >= 512 K 0 = RAM < 512 K	1 = RAM >= 2048 K 0 = RAM < 2048 K
	Ch15	LP-1/2/3 Only: 1 = extended FLASH memory, 0 = No extended FLASH memory	Read
	Ch16	1 = RAM Fault, 0= RAM OK	Read
#YSYS.ENABLE	1= First scan of ladder after downloading ladder logic or after a Toolbox *Enable Logic Processing* command is received.		Read
#YSYS.SCAN1	1 = First scan of ladder logic after a warm start, a power reset or after downloading the SDB file. Not activated when waking up from sleep mode or after downloading ladder logic.		Read
#YSYS.WARMST	1 = Warm reset CPU (Setting this bit will warm start the RTU). 0 = Normal CPU operation		Read/Write

Clock Registers

Note: continuously writing to the real-time clock registers will cause the clock to stop or behave erratically.

#YMSEC	The current milliseconds (0-999). Resolution is currently 10ms.	Read/Write
#YSEC	The current second (0-59)	Read/Write
#YMIN	The current minute (0-59)	Read/Write
#YHOUR	The current hour (0-23). 0=12 AM (midnight), 23=11 PM	Read/Write
#YWEEKDAY	Day of the week (1-7). 1=Sunday, 7=Saturday	Read/Write
#YDAY	Day of the month (1-31)	Read/Write
#YMONTH	Month of the year (1-12)	Read/Write
#YYEAR	Current year since 1900 (0-170). Eg. 1=1901, 104=2004	Read/Write

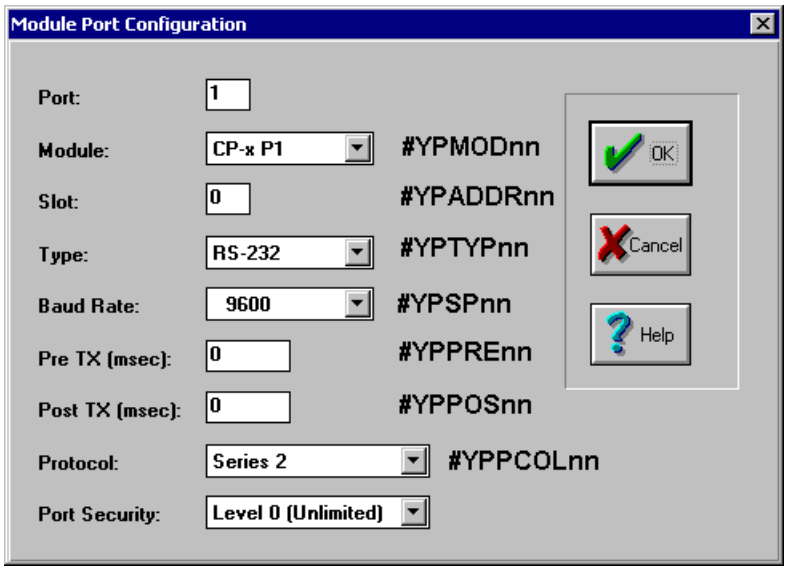
Timer Flags

These are system register bits that are periodically active for one scan of the ladder logic. Timer flags are ladder contact addresses that be used multiple times in ladder logic and are useful for triggering events periodically.

#YTICK.100TH	Hundredth of a second timer tick. Activated every 1/100 of a second (10ms). Note: if the ladder scan rate is less than 100 times a second, some of the times when YTICK.100TH is true will be missed by the ladder logic.	Read
#YTICK.TENTH	Tenth of a second timer tick - activated every 1/10 of a second (100ms)	Read
#YTICK.SEC	One second timer tick	Read
#YTICK.10SEC	Ten second timer tick	Read
#YTICK.MIN	One minute timer tick	Read
#YTICK.10MIN	Ten minute timer tick	Read
#YTICK.HOUR	One hour timer tick	Read
#YTICK.DAY	One day timer tick	Read

RTU Data - Port Registers

Port Registers contain the configuration parameters and communication statistics for the ports configured in the RTU. Most of these registers are read / write which means they can be changed while the RTU is running by using ladder logic. For the following registers, *nn* is the port number (1-16) and *cc* is the register channel where applicable.



Configuration Parameters

#YPMODnn	The type of module (PC-1/CP-x/MC-x) and port number (1-3) configured for port *nn*. CP-x P1= 30, PC-1 P1 = 45, MC-x P1 = 59, CP-x P2= 94, PC-1 P2 = 109, MC-x P2 = 123, CP-x P3= 158, MC-x P3 = 187, CP-x P4 = 222.	Read/Write
#YPADDRnn	Slot Address (1-64) of the module containing port *nn*	Read/Write
#YPTYPnn	Port Type. 0=RS232, 1=RS485, 2=Radio, 3=Pline, 4=PSTN, 5=TMR, 6=RADIO_HOT, 9=RS422, 8=Mobitex, 10=Ethernet, 11=Video, 12=RS232 RADIO, 13=MicroX RADIO, 14=GPS Internal, 15=GPS External, 16=Line-2.	Read/Write
#YPSPnn	(300 - 57600, 65535=115200 bps) Port Speed	Read/Write
#YPPREnn	Pre-Tx Delay (ms)	Read/Write
#YPPOSnn	Post-Tx Delay (ms)	Read/Write
#YPPCOLnn	Port protocol. Series 2 = 0; ADS Data Logger = 7; Allen Bradley = 21; ALERT = 71; Alstrom Relay = 50; ASCII No Parity = 43; ASCII Even Parity = 53; BCL ARC Device = 52; Conitel = 80; Cooper = 64; DATAC = 14; Datataker = 46; Datran DT300 = 58; DNP-3 (all settings) = 38; DV1000 = 34; Form 4C = 6; FUJI Micrex-F = 29; FUJI NJ Series = 36; GE CCM = 26; GE SNP-X = 47; GE T60 Relay = 51; Genisys = 44; Genisys Master = 81; GPS = 49; GPS NMEA = 76; HART = 23; IDEC PLC = 48; INLINE = 17; INLINE2 = 55; INTRAC = 16; JZA Train Control = 41; MAC-800 = 10; Mercury = 28; Microtran = 11; Mier Transposer = 62; Minitran = 12; G&F Minitran = 18; MTran Host = 33; Mbus ASCII Init = 66; Mbus init, S2 = 2; Mbus init&resp, S2 = 2; Mbus SCADA, S2 = 2; Mbus slave, S2 = 1; Mbus init&parity = 130; Mbus init&resp&par = 130; Mbus SCADA&parity = 129; Mbus slave&parity = 129; Modem Sw Unit = 15; Monitor WeatherStn = 45; Multitrode 2PC = 25; NEC DCU = 22; OMRON = 19; PEEK = 74; Remote Data Logger = 39; S1 Ctrl, Mbus, S2 = 0; S1 Outstn, S2 = 0; S1 Ctrl, no CRC, S2 = 4; S1 Out, no CRC, S2 = 4; SER_SSPA = 77; Shaft Encoder = 20; Simatic TI = 72; SM6615 = 78; STIC Gauge = 27; Tandberg = 63; Traffic Light Controller = 61; TRIO Eseries = 70; TS5000 = 30; User-Defined (all settings) = 57; YSI Logger = 42.	Read/Write

Communication Statistics

#YPERRnn	Port error. 1=Error. For GPRS, 0=modem initialised successfully		Read
#YPINACnn	(0-60,000) Inactivity Timer. The number of seconds since the last valid message received on port *nn*. If the timer reaches 60,000 seconds it will stop incrementing.		Read
#YPRTUnn	Online RTU address (for PSTN). Returns the RTU address (1-249) of the remote RTU that the local RTU is currently connected to via PSTN. For GPS_NMEA protocol (for a GPS device), used to set which network registers store the GPS information. Eg if #YPRTU2 is set to 5, the GPS information will be stored in the network registers for RTU5 ie #NR5.xx. Please refer to the GPS_NMEA protocol document for more information.		Read
#YPRXCnn	Received Characters. The number of unread bytes/characters currently in the receive buffer of port *nn*. This parameter is used with the Rx User ladder block to determine when to read the port buffer.		Read
#YPSIGnn	(0-65535) Signal strength. Device dependent. For GPRS, returns the signal strength and quality as a single number - Result=Signal strength x 100 + Signal quality. Eg. If #YPSIGnn=1204 (decimal), then the signal strength is 12 and the signal quality is 4. Note: 65535 = invalid response.		Read
#YPSTnn.cc	Port status		
	Ch1	Reserved	
	Ch2	1 = Message waiting for a reply on port *nn*. Used to check that a port is free before initiating a new message from that port. Please see the topic Example – Sending the Exception Report.	Read
	Ch3	1 = Port online (PSTN modem and TMR only). Please see the topic Configuration, Port List, Pre Tx for more information.	Read
	Ch4	1 = Off Hook (TMR only)	Read
	Ch5	1 = Calling / incoming call (TMR only)	Read
	Ch6	1 = In Service (TMR, Mobitex radio and GPRS only) For GPRS, 1=modem connected to the network in listen mode.	Read
	Ch7	1 = Link Active (Ethernet T option board, TMR, Mobitex & DV1000 only). Note: supported by Ethernet T option boards version 1.1 and newer (port is labeled *E'NET-T 1.1*).	Read
	Ch8	1 = Last Dial Failed. Only applicable to PSTN. Set whenever a dial fails on the port and remains set until a dial is successful on the port. Last Dial Failed is cleared after a warm start.	Read
	Ch9	1 = Block message pending. A block message generates other *child* messages to retrieve the information required. A block message can generate hundreds of child messages and be active for minutes. The block message bit is set while the block message is still in progress. If a block message has multiple target RTUs then the block message pending bit is referenced to the port (and RTU address) of the first RTU in the list. It is recommended that only one block message be initiated at any one time. This prevents overloading of the message buffer and prevents the RTU toggling between the child messages of the block messages. While a block message is in progress, the child messages still cause the message pending bits for the port and network link to be set and reset.	Read
	Ch10	1 = Port error (Mobitex radio only)	Read
	Ch11	1= Initialise GPRS modem.	Write
	Ch12	0 = Use fixed UDP destination port based on port protocol [default] 1 = Use dynamic UDP destination port. Required when using Citect 6.x+. Note: this parameter is not supported by MC-10/11/12 Ethernet ports.	Read/Write
	Ch14	1= CTS Active. Eg: The CTS pin on the port can be wired to the ring indicator on a modem to monitor an incoming call.	Read
	Ch15	Character error. RTU use only	Read
	Ch16	1 = Data Carrier Detected	Read

RTU Data - Network Link Registers

Network Link Registers contain the configuration parameters and communication statistics for the network links to each outstation RTU. Most of these registers are read / write which means they can be changed while the RTU is running by using ladder logic. For the following parameters *rrr* is the destination RTU address (1-249) and *cc* is the register channel (1-16) where applicable.

Configuration Parameters

#YLSIDrrr	System ID	Read/Write
#YLDIRrrr	Direct or indirect link. Returns an integer value. 1 = Direct Connect (via port), 0 = Indirect (via RTU).	Read/Write
#YLVIArrr	Via port number or RTU address (1-16 or 1-249 respectively)	Read/Write
#YLRETRIESrrr	(1-255) The number of retries for each message	Read/Write
#YLTOUTrrr	Comms timeout (ms)	Read/Write
#YLIPADLrrr	(0-65535) Lower half of IP address *A.B.C.D*. Chs 1-8=C, Chs 9-16=D. To set C and D, use #**YLIPADL**rrr=C+256xD.	Read/Write
#YLIPADHrrr	(0-65535) Higher half of IP address *A.B.C.D*. Chs 1-8=A, Chs 9-16=B. To set A and B, use #**YPIPADH**rrr=A+256xB.	Read/Write

Communication Statistics

#YLFCrrr *	Comms attempt fails since last success (0-65535)		Read/Write
#YLSUCCrrr * [#]	Counter of comms successes (0-65535) since last reset. The success counter is incremented when an exception report is received or a reply is received to an initiated message. An RTU does not increment the success counter when relaying (store and forwarding) a message to another RTU.		Read/Write
#YLFAILrrr * [#]	Counter of comms *attempt* failures (0-65535) since last reset. The fail counter is only incremented when there is no reply to an initiated message (or the message has timed out). An RTU does not increment the fail counter when relaying (store and forwarding) a message to another RTU.		Read/Write
#YLSTrrr.cc *	Link Status		
	Ch1	1 = Last message failed (all attempts failed). Cleared when a reply is received to an initiated message or an unsolicited message is received eg. the RTU is polled by a remote RTU or an exception report is received.	Read
	Ch2	1 = Message waiting for a reply from RTU *rrr*. Useful for Ethernet ports. Note: when using a radio, private line or serial port to communicate with 2 or more RTUs #YPSTrrr.2 should be used instead of #YLSTrrr.2 to avoid message clashes. Please see the topic Example – Polling Data.	Read
	Ch3	1 = RTU online (PSTN modem only). Please see the topic Configuration, Port List, Pre Tx for more information.	Read
	Ch5	1 = Dialling (PSTN modem only)	Read
	Ch8	1 = Last dial failed (PSTN modem only)	Read
	Ch9	1 = Block message pending. Please see #YPSTnn.cc Ch9 for details.	Read
	Ch11	1 = Close socket connection (CP-11 or MC-11 ports only). When set, forces the RTU to close the existing socket connection for a remote RTU. When a new message is initiated, the new network link settings will then be used (useful for creating a redundant comms path).	Read/Write
	Ch13	1 = Last *initiated* message failed. Same as #YLSTrrr.1 but only set or reset after an initiated message. This parameter is only updated when used with the Kingfisher or DNP3 protocols.	Read
#YLUPDCrrr	The number of times the network data has been updated since last reset. Each time a network register is changed, the update counter is incremented. After receiving a block of data values (eg. from an exception report or as a reply to a poll message), the update counter will be incremented once for *each new* data value in the block of data.		Read/Write
#YLLOGIDXrrr	(0-65535) Event log current index pointer. Points to the last log in RTU *rrr* that was uploaded by the local RTU.		Read/Write
#YLPENDINGrrr	The number of messages pending (waiting for a reply) for RTU *rrr*.		Read

* These parameters are reset to 0 after a warm start or after a power up. Communication fail counters are incremented for each message attempt. If all dialling attempts fail, a communication fail is also recorded.

[#] DNP3 Protocol: the success counter is incremented when a message is received. If the message is a multi-fragment message (more than 2048 data bytes have been requested) then the RTU will reply with two or more fragments of data and will expect to receive an ACK after each fragment (except the last one). A success is then recorded for each ACK received or a fail is recorded if no ACK is received.

RTU Data - Module Registers

Module Registers contain the module type number and monitor code version of each module in the RTU.

For the following registers, *ss* is the slot address (1-64) of the module.

#YMTYPEss	Module type (1-255) 1 AI-1 / AI-4 2 AO-2 3 RT-1 5 DI-1 6 DI-5 7 DO-1 8 DO-2 / DO-5 / DO-6 9 DI-10 10 IO-1 11 IO-2 12 IO-3 13 SBX / IO-3 14 IO-4 15 AO-3 19 AI-10 25 LM-2 26 RD-1 29 CP-x slave (for redundancy) 30 CP-10/11/12/21 master 31 PSU-1, PS-1, PS-10/11/12/20/21/22 45 PC-1 46 CP-12 Master 57 LM-1 59 MC-x 255 No module	Read
#YMVERss	The software version number of the IO module. Each IO module has a microcontroller (8051) that is programmed with the current monitor code and labeled with a version number. YMVER returns an integer value in the range 0-255 where 0 is the oldest version.	Read

Appendix – Potential DEP Issue on Windows Vista, Windows 7 or Windows Server2008

(May also apply to other Windows versions)

Microsoft introduced Data Execution Prevention (DEP) into Windows XP SP2. DEP closes down a program that it suspects has malicious content.

This feature has the potential to prevent Toolbox32 from executing under the false assumption it contains malicious content.

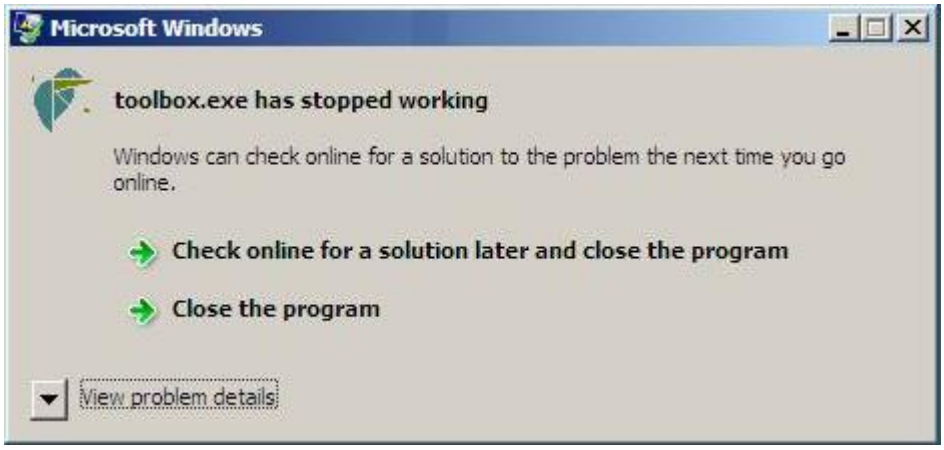
In most installations this is unlikely to occur, however if it does occur when attempting to run Toolbox32, then the following link from Microsoft describes the steps to configure Toolbox32 to execute normally.

http://windows.microsoft.com/en-AU/windows-vista/Change-Data-Execution-Prevention-settings

The procedure is also described in detail below.

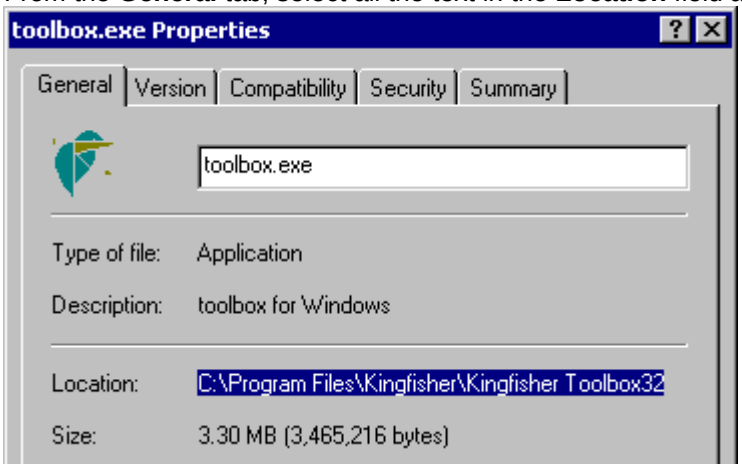
*** A copy of this note is stored in the Toolbox32 execution directory, should it be required for future reference. It may also be found in the Toolbox32 help.

DEP may generate the following error message when Toolbox 32 is run (as observed in Windows Server 2008):

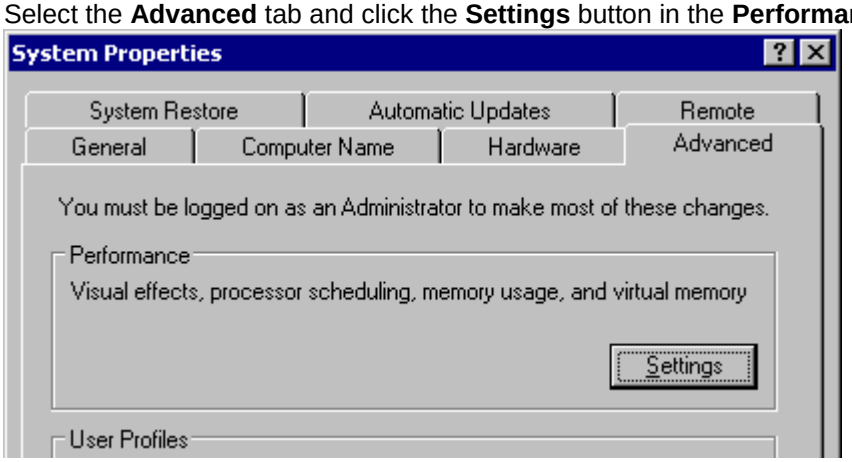

Microsoft Windows
toolbox.exe has stopped working
Windows can check online for a solution to the problem the next time you go online.
Check online for a solution later and close the program
Close the program
View problem details

How To Run Toolbox.exe

1. Using **Windows Explorer**, locate **Toolbox.exe** (after software has been installed by running SETUP.EXE). RIGHT click on the file and choose **Properties**. Toolbox is located in **C:\Program Files (x86)\Kingfisher\Kingfisher Toolbox32** by default.
2. From the **General** tab, select all the text in the **Location** field as shown below for **Toolbox.exe**



3. Press Ctrl + C on the keyboard to copy the text.
4. Right click the **My Computer** icon on the desktop and choose **Properties**.
5. Select the **Advanced** tab and click the **Settings** button in the **Performance** section as shown below.



6. A window will appear entitled **Performance Options**. Click the **Data Execution Prevention** tab and the window below will appear.

Performance Options
Visual Effects
Advanced
Data Execution Prevention
Data Execution Prevention (DEP) helps protect against damage from viruses and other security threats. How does it work?
Turn on DEP for essential Windows programs and services only
Turn on DEP for all programs and services except those I select:
Add...
Remove
Your computer's processor does not support hardware-based DEP. However, Windows can use DEP software to help prevent some types of attacks.
OK
Cancel
Apply

7. Select the option "**Turn on DEP for all programs and services except those I select:"**
8. Click the **Add** button that is no longer greyed out and a standard Open File window will appear. Click in the **File Name** box and then press **Ctrl + V** on the keyboard to paste the location of **Toolbox.exe** into the box.

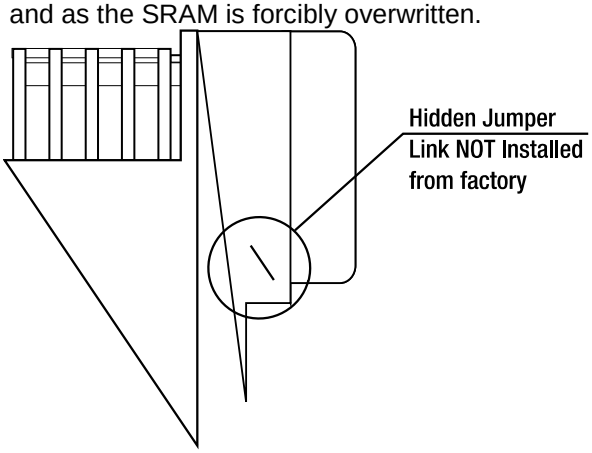
9. Click the **Open** button and the window will close and the software will now be visible in the DEP exception list. Then click the **Apply** button and close all the open windows.
10. **Toolbox 32** should now run normally.

Appendix – Clearing Configuration with Monitor Mode

Behind the LED lens of the CP-12 there is a diagonally mounted jumper. This is a hardware feature unique to the CP-12 processor.

When the jumper pins are shorted while the module is being reset (or power cycled), the processor will enter Monitor Mode (the Kingfisher bootloader) and perform a read/write test of the SRAM. This will forcefully clear SRAM, having the same effect as removal of the SRAM link on the back of the CP-12 module.

Monitor Mode is a deterministic and efficient way of clearing a Processor as the module does not need to be removed from the Backplane, and as the SRAM is forcibly overwritten.



Note: The RAM test is extremely fast. The CP-12 will clear the RAM nearly instantly. To exit monitor mode, remove the short and power cycle the unit again.

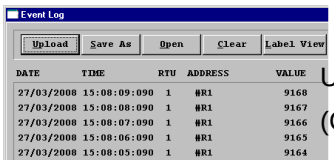
LP-2/3 Low Power RTU

KINGFISHER

1. Introduction

LP-2/3 Features

Stand-alone RTU



Up to 300,000 event logs

(Can view using Toolbox 32)

Configured using Toolbox 32 software

Ready to run!

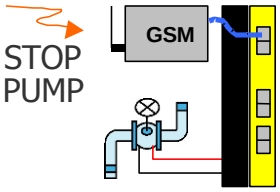
All voltages and ports enabled by default

Two serial ports and one (LP-2) or two (LP-3) option ports

Very low power consumption

Isolated serial port 2 supports RS422/RS485

Eight digital inputs



Remotely control LP-2/3 outputs using a mobile phone

(LP-2/3 requires a GSM)

Two digital outputs

(Relay)

Receive SMS alarms from the LP-2/3 containing status values and variables

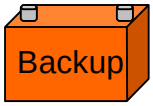
(LP-2/3 requires a GSM or PSTN modem)

Up to four digital inputs can be configured to operate as transistor digital outputs

5 /12 / 24 V Outputs for powering I/O or an external device

All controllable by logic

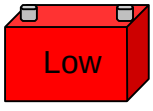
Four analog inputs



Can connect a backup battery

(LP-2/3 provides trickle charging)

One analog output (LP-3 only)



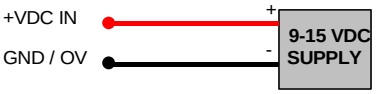
Low voltage shutdown (software configurable)

2. Quick Start

An LP-2/3 is configured using Toolbox 32 software as detailed below.

1. Power up the LP-2/3 by connecting a suitable DC supply.

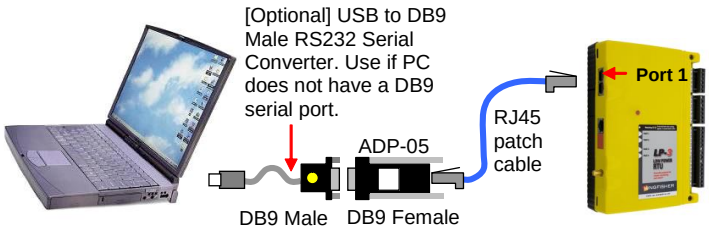
The red light (LED) should start blinking once per second if the LP-2/3 is running correctly.



Watchdog LED: 1 second flash = RUNNING OK

2. Connect the Toolbox 32 cable* to port 1 of the LP-2/3 as shown.

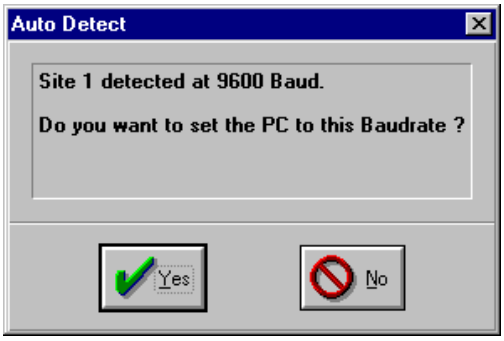
* For ADP-05 wiring details, please see the topic Specifications, Communications Ports.


[Optional] USB to DB9 Male RS232 Serial Converter. Use if PC does not have a DB9 serial port.
ADP-05
RJ45 patch cable
Port 1
DB9 Male
DB9 Female

3. After starting Toolbox 32, ensure the correct PC communications port is being used by the software (usually COM1).

The COM port can be changed from the Toolbox 32 menu - **Configuration, PC Setup.**

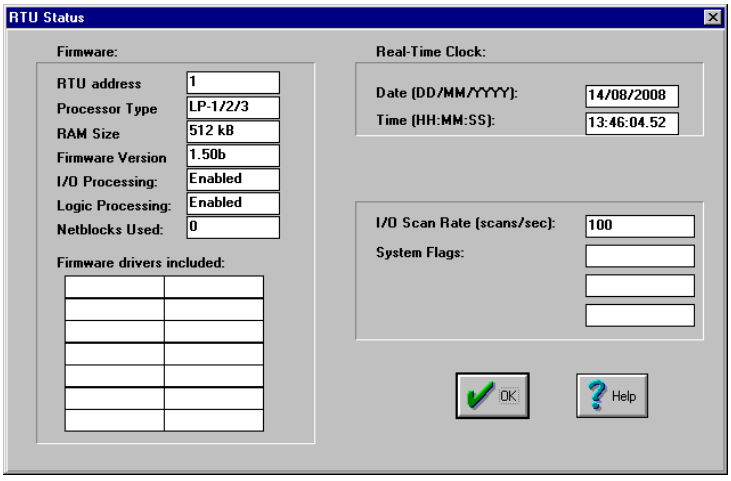
The address and baudrate of the LP-2/3 can then be detected using **View, Auto Detect**.



4. Check the LP-2/3 clock is set to a valid time and date using **View, RTU Status**.

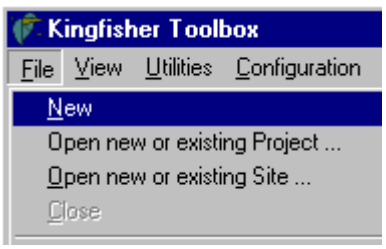
The time and date can be changed from **Utilities, Set Real-Time Clock**.

Warning: the LP-2/3 will behave unpredictably until the clock contains a valid setting.



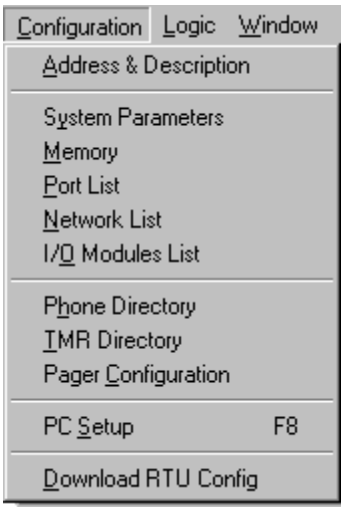
5. Create a new configuration by selecting **File, New**.

The new configuration can be saved by selecting **File, Save As** and specifying a different name or the default name (NEW.SDB) can be used.

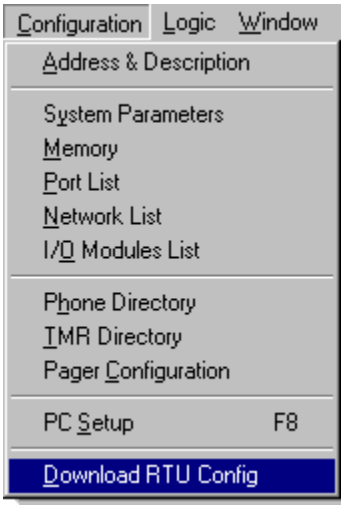


6. Configure the following settings from the **Configuration** menu:

- **Address & Description** – Set *Site Address* to a number in the range of 1 to 249
- **System Parameters** – Set *RTU/CPU Type* to *LP-1/2/3*
- **Memory** – Set *Event Logs* to 64K, *Compiled Logic* to 32K, *Network Reg Blocks* to 2K and *Firmware Drivers* to 32K (other values can be used). Note: up to 300,000 event logs can be stored in the RTU if *LP1/2/3 Exp. Memory* is ticked. Event logs stored in expanded memory can only be uploaded using Toolbox 32.
- **Port List** – Select button **2** (for port 2). Set *Module* to *CP-x P2*.
- If a PSTN modem, Radio, GSM, RS485 communications or an option board is being used, please see the Port Configuration chapter for more essential settings.



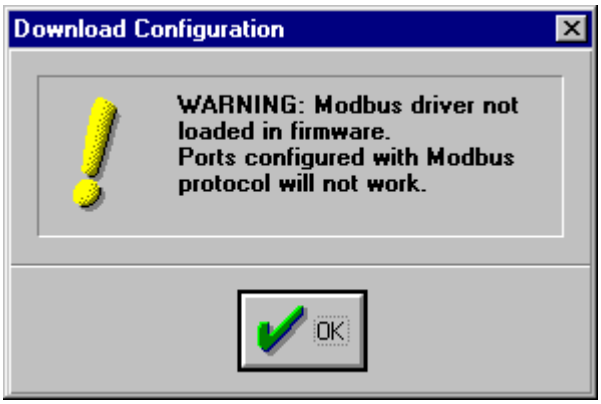
7. Download the configuration file into the LP-2/3 using **Configuration, Download RTU Configuration**.



8. When firmware drivers are required (eg. for paging, Modbus, DNP3), they are always downloaded into SRAM (battery backed memory).

After an RTU configuration is downloaded into the LP-2/3 that allocates memory for firmware drivers, the drivers can then be downloaded.

LP-1/2/3 drivers use the file extension **DHI** and are available from http://helpdesk.servelec-semaphore.com


Download Configuration
WARNING: Modbus driver not loaded in firmware.
Ports configured with Modbus protocol will not work.
OK

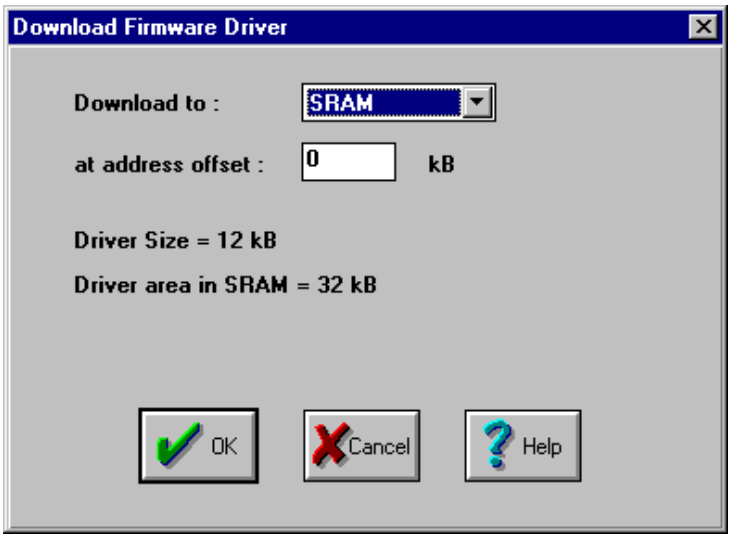
9. A firmware driver is downloaded using **Utilities, Advanced, Download Firmware Driver.**

Select *Download to* **SRAM** *at address offset* **0** kB.

For more than one driver, ensure that they do not overlap in memory ie. use an *address offset* that is equal to (or greater than) the size of all the drivers already in the RTU.

When using the Paging driver, the RTU configuration will need to be downloaded again (step 7 above) as the paging settings are only downloaded when the paging driver is present in the RTU.

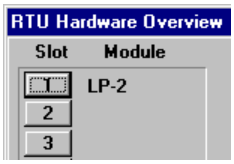
Note: if the LP-2/3 is cold-started, the RTU configuration and firmware drivers are cleared and will need to be downloaded again.

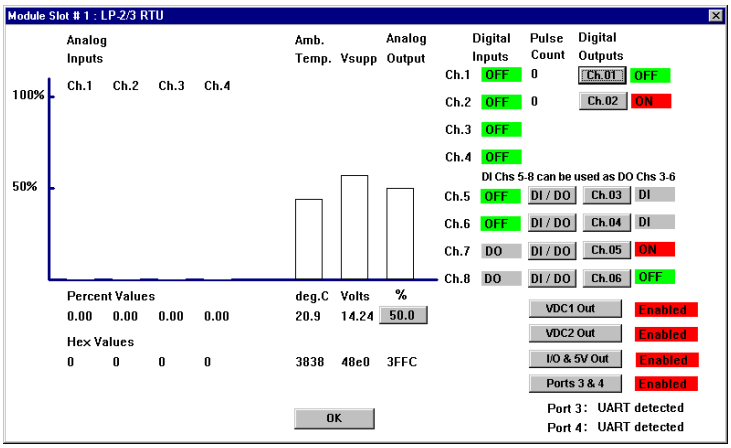


10. LP-2/3 hardware options are set from **View, Hardware Overview.**

By selecting the Slot **1** button, a number of settings are displayed as shown.

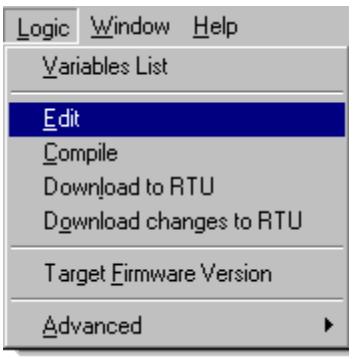
- Ensure **I/O & 5V Out** is enabled (default setting)
- Ensure **V1 Out** is enabled (default setting) if using V1 Out to power I/O or an external device
- Ensure **24V Out** is enabled (default setting) if using 24V Out to power I/O or an external device
- Digital inputs 5 to 8 can be changed into digital outputs 3 to 6 by selecting the corresponding **DI /DO** button
- Ensure **Ports 3 & 4** are enabled (default setting) if using a comms option board





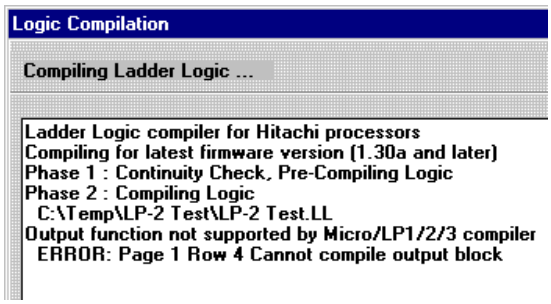
11. To add ladder logic functionality (including low power operation), please see the Ladder Logic Examples chapter.

Note: if any hardware settings were changed in the hardware overview (step 10 above), these settings should be included in ladder logic as the LP-2/3 will revert to the default settings after a cold-start (please see topic - Ladder Logic Examples, Setting Hardware Options Using Ladder Logic).



After downloading any ladder logic, the LP-2/3 is now configured!

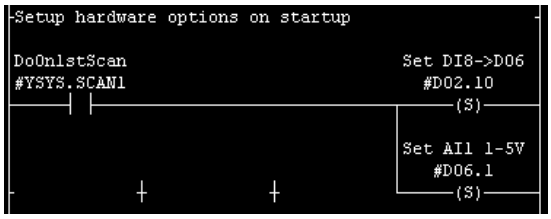
3. Ladder Logic Examples


Logic Compilation
Compiling Ladder Logic ...
Ladder Logic compiler for Hitachi processors
Compiling for latest firmware version (1.30a and later)
Phase 1 : Continuity Check, Pre-Compiling Logic
Phase 2 : Compiling Logic
C:\Temp\LP-2 Test\LP-2 Test.LL
Output function not supported by Micro/LP1/2/3 compiler
ERROR: Page 1 Row 4 Cannot compile output block

Logic Limitations

PID, Clock Synchronization, **Report Printer** and indirect addressing in function blocks are not supported by an LP-2/3.

Registers #R1921 to #R2048 should be kept free to allow the use of firmware diagnostics.



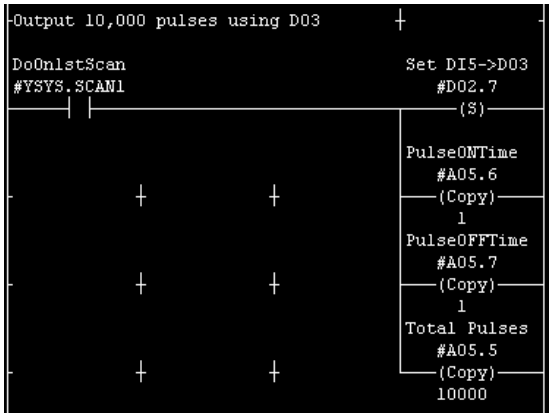
The logic shown configures digital input 8 to be a digital output and configures analog input 1 for live zero 1-5 V (or 4-20 mA).

Setting Hardware Options Using Ladder Logic

After a cold start or firmware download, hardware options are reset to their default state so it is advisable to configure any custom settings in ladder logic.

The hardware options that can be set are listed in the topic **LP-2/3 Registers**, **Hardware Setup Registers**

Output voltages **5V Out**, **V1 Out** and **24V Out** can be individually controlled using ladder logic.



High Speed Pulse Output (Digital Output 3)

Digital output 3 (when enabled) can be configured as a high speed pulse output (500 Hz maximum - depends on pulse settings)

The number of pulses to output is configured in #AO5.5

Pulse ON time (x1 ms) is configured in #AO5.6

Pulse OFF time (x1 ms) is configured in #AO5.7

The value in #AO5.5 is decremented by the RTU as each pulse is outputted

Setting Up Low Power Operation *

In addition to controlling output voltages, the LP-2/3 can be put to sleep. This results in very low power consumption. The LP-2/3 can also count pulses (from digital inputs 1 and 2) while remaining in subactive sleep mode.

In Sleep mode:

- LED flashes every four seconds
- Analog inputs are disabled
- Analog output (LP-3 only) is reset to zero
- Digital outputs 1 and 2 (relay) hold last state
- Ports 1 to 4 are disabled (a port 1 CTS change can still wake the RTU)
- **5V Out**, **V1 Out** and **24V Out** are disabled
- The CPU is placed in sleep mode for a configurable time interval (IO and communications are no longer processed and ladder logic is stopped)
- Pulses can be counted from digital inputs 1 and 2 while remaining in Subactive sleep mode. Note: **CPU 5V Out** is still enabled during Subactive sleep mode and can be used to power digital inputs 1 and 2.

The RTU will wake up when:#

- The sleep time has expired
- A serial cable (with a CTS/RTS loop) is connected or removed from port 1 (default setting)
- Digital input 1 or 2 become active (if enabled before entering sleep mode)

Configuration:

Sleep mode is configured using the two parameters **#YPDTIME** and **#YPDSTAT**. **#YPDSTAT** is defined differently for LP-x RTUs than for other types of Kingfisher RTUs as follows:

#YPDSTAT	Wakeup Conditions (integer read/write)
Ch 1	Time wakeup (default setting. Always set when #YPDTIME is set)
Ch 2	Digital input 1 change of state
Ch 3	Digital input 2 change of state
Ch 4	Not Used
Ch 5	Port 1 CTS change (default setting)
Chs 6-15	Not Used
Ch 16	Subactive sleep mode. The LP-2/3 remains in sleep mode while counting pulses from digital inputs 1 or 2. Eg. Can enter subactive sleep mode by writing 16#8010 (8010 Hex) to #YPDSTAT (this will allow the LP-2/3 to count pulses and will also enable wakeup on port 1 CTS change). **5V Out**, **V1 Out** and **24V Out** are also disabled in this mode.

An RTU is put to sleep by setting #YPDTIME (seconds of power down). The LP-x will then wakeup when the sleep time has expired or port 1 has a CTS change by default. Other wakeup conditions can be used by writing to #YPDSTAT before setting #YPDTIME. When woken up, the RTU clears #YPDSTAT and then sets the channel corresponding to the wakeup event (#YPDTIME is also cleared on wakeup).

* An LP-2/3 does not use the **IO Power Saving Control** settings of the System Parameters (as configured in **Configuration, System Parameters**) and must be configured for low power operation using ladder logic.

If the above wakeup conditions have not been set, the RTU can be interrupted from sleep mode by powering down the RTU (by disconnecting the input supply) and removing the SRAM battery link (LK1) for 5 minutes (this will clear the RTU's configuration).

The examples below show how RTU2 (the LP-2/3) is put to sleep for 580 seconds (almost 10 minutes), 2 seconds after a successful message to master RTU1. RTU2 will wake up after 580 seconds or if digital input 1 or 2 becomes active or if a serial cable is connected or disconnected from ports 1 or 2. If RTU2 is woken up by digital input 1 or 2 and an exception report occurs, the RTU will go back to sleep 2 seconds later. Master RTU1 can be configured to poll RTU2 after 600 seconds of quiet time (ie since the last exception report received). When master RTU1 polls RTU2, RTU2 will have just woken up and after the poll, will go back to sleep again. This setup ensures the data at master RTU1 is never older than 600 seconds and ensures that RTU2 spends a lot of time sleeping.

CAUTION: an Edge trigger (or Change block) should not be used to put the LP-2/3 to sleep. Before scanning the ladder the first time (or after a wakeup or warm start) the LP-2/3 clears its last state internal registers. An edge trigger will then always be true on the first scan – possibly sending the LP-2/3 into a continuous wake/sleep cycle.

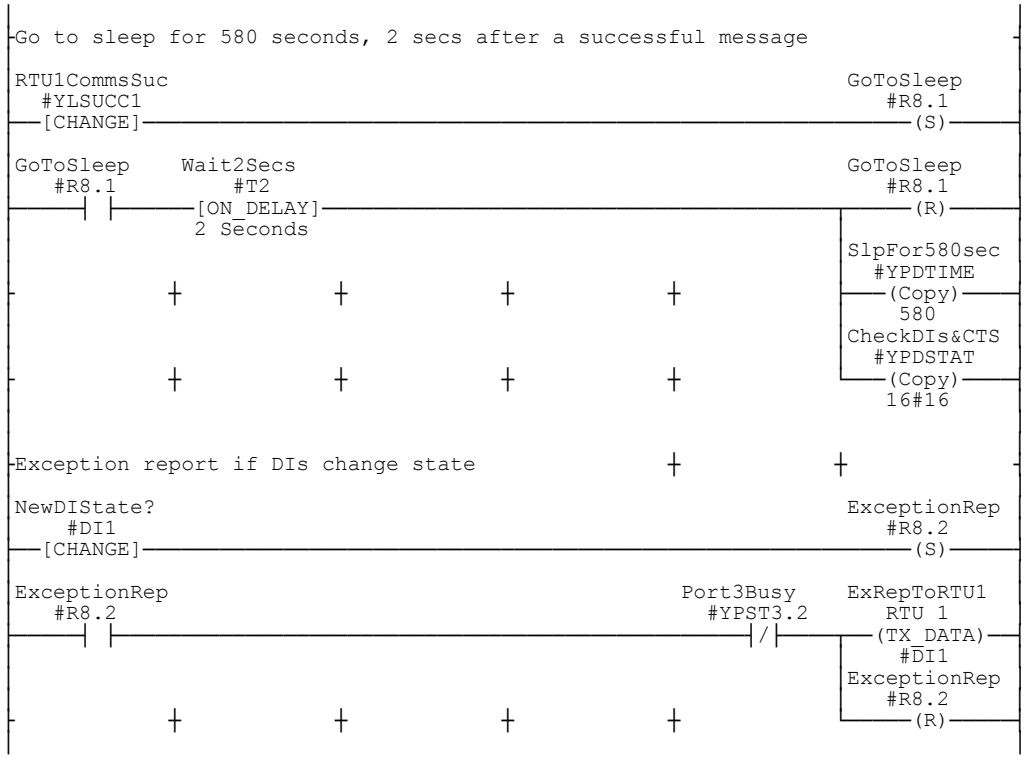


Figure: Example LP-2/3 Low Power Sleep Mode For Outstation RTU2

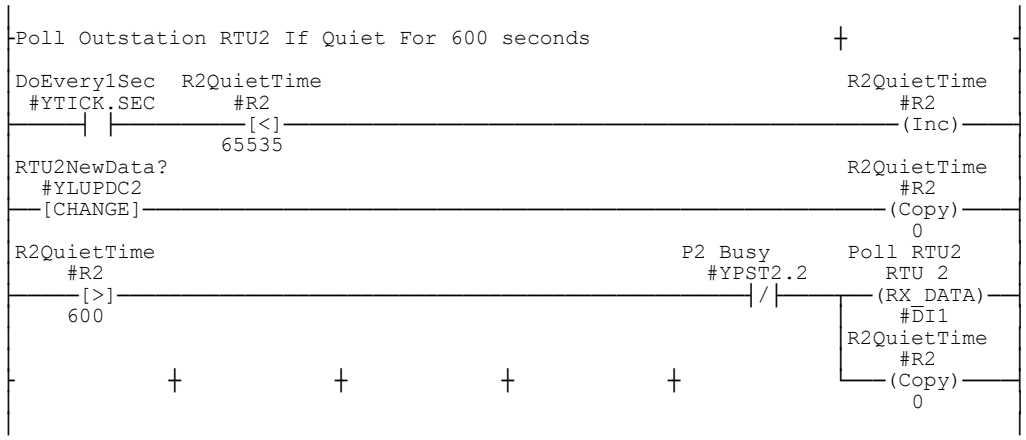
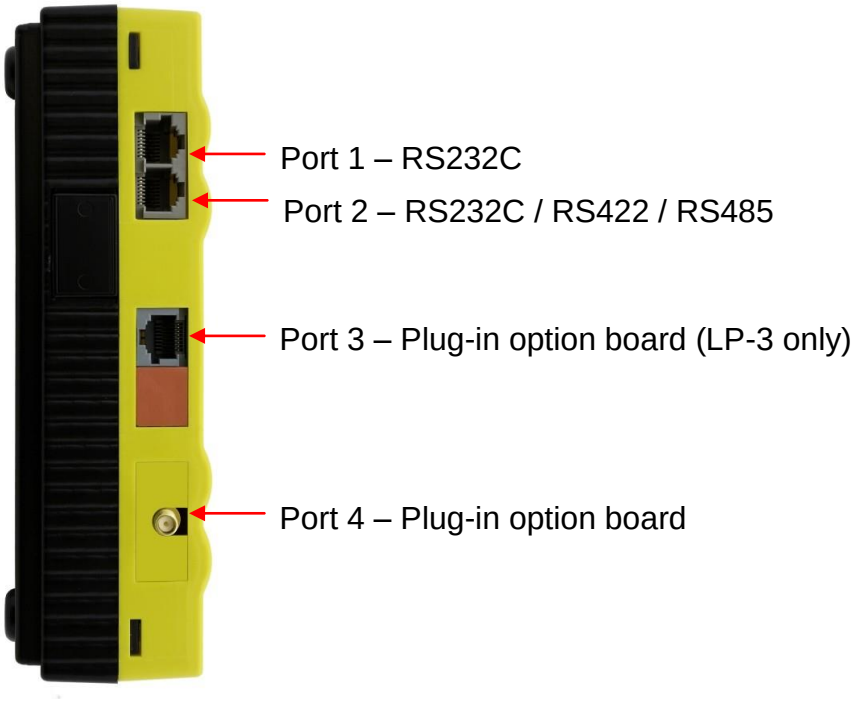


Figure: Example Master RTU1 Ladder Logic

4. Port Configuration



Note: All four ports are automatically configured as RS232, 9600 bps after a cold start.

Port 1 and Port 2 Configuration

Note: Pre and Post TX delays have a resolution of 10ms (eg 10ms will be 0-10ms delay).

Port	1 or 2	
Module	CP-x P1 or CP-x P2	
Slot	0 or 1	
Type	PSTN, RS232, RS422 (Port 2 only), RS485 (Port 2 only)	
Baud Rate	300 to **38400**	
Pre/Post TX (ms)	PSTN or RS232	RS422/RS485 (Port 2 only)
	0/0	50/50 (300 Baud) 30/30 (600 Baud) 20/20 (1200-38400 Baud)

Port 3 (LP-3 only) and Port 4 Configuration

Port	3 or 4					
Module	CP-x P3 or CP-x P4					
Slot	0 or 1					
Type (depends on installed option board)	RS232 or PSTN	RS485 or RS422	Line-2 Private Line	Line-2 Analog Radio	RS232 - Fibre Optic	SS Radio
Baud Rate	300 to 115200	300 to 115200	1200	1200	300 to 115200	9600 (9XTend) 19200 (24XStream)
Pre/Post TX	0/0	20/20	100/50	300/100	0/0	0/0

5. Backup Battery Calculations

This chapter allows the size of the backup battery to be determined depending on how long the LP-2/3 is required to run without an active supply (from mains or solar).

A 12.0 VDC power supply was connected to the battery input terminals on the LP-2/3 (version 1.3b PCB) and the current draw for each hardware setup was measured.

LP-2/3 Current Consumption (mA @12 VDC)	
Sleep mode OR Subactive sleep mode, ladder and IO disabled (except for DI1 or DI2 powered by CPU 5V), switchable voltages and ports 3 & 4 disabled	2
Run mode, ladder running, IO disabled (except for DI1 and DI2 powered by CPU 5V), no option boards installed, switchable voltages disabled (**5V Out**, **24 V Out and V1 Out**), ports 3 & 4 disabled	40
I/O & 5V Out enabled (must be enabled to read I/O except for DI1 and DI2 powered by CPU 5V)	+8
V1 Out enabled (VDC1 Out)	+0
24V Out enabled (VDC2 Out)	+5
Ports 3 & 4 enabled (with port 3 and/or 4 port configured and no option boards installed)	+4
Digital input ON	+0.4
Digital output CLOSED	+0
Analog input 5 V (100%)	+0
Analog input 20 mA (100%) with external 250 Ohm resistor installed, powered by LP-2/3 24 V	+56
Analog input 20 mA (100%) with external 250 Ohm resistor installed, powered EXTERNALLY	+0
Analog output 5 V (100%) with external high impedance load	-1
Analog output 20 mA (100%) with external 250 Ohm resistor installed, powered by LP-2/3 24 V	+49
Analog output 20 mA (100%) with external 250 Ohm resistor installed, powered EXTERNALLY	-1
Isolated Serial Option Board (**5 V Out** and port must be enabled)	+28
Dial (PSTN) Option Board (**5 V Out** and port must be enabled)	+65
Line-2 Option Board (**5 V Out** and port must be enabled)	+4
Fibre Optic Option Board (**5 V Out** and port must be enabled)	+11
2.4 GHz (50 mW) Spread Spectrum Radio Option Board, Receive (**5 V Out** and port must be enabled)	+20
2.4 GHz (50 mW) Spread Spectrum Radio Option Board, Transmit (**5 V Out** and port must be enabled)	+22
900 MHz (1 W) Spread Spectrum Radio Option Board, Receive (**5 V Out** and port must be enabled)	+46
900 MHz (1 W) Spread Spectrum Radio Option Board, Transmit (**5 V Out** and port must be enabled)	+122

Average Current Consumption

The average current consumption is calculated by multiplying the fraction of time that the LP-2/3 is in each mode by the current consumption of that mode. These currents are then totaled to provide the average current consumption.

Example: An LP-2/3 has one 900 MHz Spread Spectrum Radio option board. The LP-2/3 continuously sleeps for 9 minutes, awakes for 1 minute (with all output voltages and port enabled) and then sleeps again. While awake it transmits for 1 second (1/60=0.0167 minutes). The LP-2/3 also has two digital inputs (assume both are active) and one current analog input (assume worst case of 20 mA / 100%).

The average current consumption (during each 10 minutes and therefore during the entire day) is:

State	**Time In State**	**Current Draw (mA)**	**Weighted Current Draw**
Sleeping	9 mins every 10 mins	2	9/10 x 2 = 1.8
LP-2/3 running with voltages and ports enabled	1 min every 10 mins	40 + 8 + 5 + 4	1/10 x 57 = 5.7
Two digital inputs ON	1 min every 10 mins	2 x 0.4	1/10 x 0.8 = 0.08
One analog input @ 20mA	1 min every 10 mins	56	1/10 x 56 = 5.6
Radio transmitting	0.0167 mins every 10 minutes	120	0.0167/10 x 120 = 0.20
Radio receiving	1 – 0.0167 mins every 10 mins	44	(1-0.0167)/10 x 44 = 4.3
		Average Current Consumption:	**18 mA** (rounded up)

Battery Backup

For maximum battery life, a maximum discharge depth of 50% is typically used. Eg. for a 7 AH (Amp Hour) battery, there is 3.5 AH of backup power. Therefore the battery can supply 3.5 Amps for 1 hour or 1 Amp for 3.5 hours etc.

The minimum size of the backup battery required depends on how long the LP-2/3 needs to be able to run without mains or solar power.

Min. Backup Battery = (LP-2/3 Average Current Consumption [mA] / 1000) x Backup Time (hours) x 2 AH

Therefore, for the above example, the minimum backup battery required to power the LP-2/3 for two days (48 hours) of no power (or adequate sunlight) is:

Minimum Backup Battery = (18 / 1000) x 48 x 2 = 1.8 AH (rounded up)

6. LP-2/3 Registers

Note: Only read/write hardware registers listed here should be written to. Other hardware registers are used by the LP-2/3 firmware and care must be taken not to overwrite these.

I/O Registers

Register	Description	Comments	Read/Write
#DI1.1	Digital input 1	0V or Open =OFF. Pulse input 1	Read
#DI1.2	Digital input 2	0V or Open =OFF. Pulse input 2	Read
#DI1.3	Digital input 3	0V or Open =OFF.	Read
#DI1.4	Digital input 4	0V or Open =OFF.	Read
#DI1.5	Digital input 5	0V or Open =OFF. Can be used as output DO2.3	Read
#DI1.6	Digital input 6	0V or Open =OFF. Can be used as output DO2.4	Read
#DI1.7	Digital input 7	0V or Open =OFF. Can be used as output DO2.5	Read
#DI1.8	Digital input 8	0V or Open =OFF. Can be used as output DO2.6	Read

#DO2.1	Digital output 1	Latched relay output (1 = closed)	Read/Write
#DO2.2	Digital output 2	Latched relay output (1 = closed)	Read/Write
#DO2.3	Digital output 3	Optional Transistor output (1 = closed ie. sink to 0V)	Read/Write
#DO2.4	Digital output 4	Optional Transistor output (1 = closed ie. sink to 0V)	Read/Write
#DO2.5	Digital output 5	Optional Transistor output (1 = closed ie. sink to 0V)	Read/Write
#DO2.6	Digital output 6	Optional Transistor output (1 = closed ie. sink to 0V)	Read/Write

#AI1.2	Analog input 1	12bit, 0-32760 = 0-100%	Read
#AI1.3	Analog input 2	12bit, 0-32760 = 0-100%	Read
#AI1.4	Analog input 3	12bit, 0-32760 = 0-100%	Read
#AI1.5	Analog input 4	12bit, 0-32760 = 0-100%	Read
#AI2.4	Pulse counter 1	0-65535 pulses. Write 0 to AO2.4 to reset	Read
#AI2.5	Pulse counter 2	0-65535 pulses. Write 0 to AO2.5 to reset	Read

AO5.4	Analog Output 1	15 bit, 0-32767 = 0-100%. LP-3 Only.	Read/Write

Power Supply Data

Register	Description	Comments	Read/Write
AI1.7	Ambient Temperature $Ambient\ Temperature = (\frac{AI1.7}{20475} - 0.6) \times 100$	0-32760 = -60 to 100°C*, Accuracy: ± 3%	Read
AI1.8	DC Supply Voltage to LP-2/3 **Mains Fail**: When powered using an AC/DC supply (@13.8 VDC) and a backup battery, If the supply voltage drops, it can be assumed that the LP-2/3 is now running off the battery and mains fail has occurred. ie: If #AI1.8>21948 (13.5V) = Mains OK If #AI1.8<21135 (13.0V) = Mains Fail	0-32760 = 0 to 20.15 V, Accuracy: ± 2% The DC Supply Voltage is either the supply voltage connected to +VIN or +B (whichever is greater) Note: **5V Out** must be enabled to allow the LP-2/3 to monitor the supply voltage.	Read
AO2.9 Chs 1-8	Low Voltage Wakeup Conditions. The initial setting in this register specifies which wakeup triggers to enable. When the RTU wakes up (after the supply voltage is restored), the wakeup trigger that was activated is written to this register by the RTU. Ch1 Time wakeup (automatically set) Ch 2 Digital input 1 Change Of State Ch 3 Digital input 2 Change Of State Ch 4 Not used Ch 5 Port 1 CTS change (default) Ch 6-8 Not used	Default: #AO2.9 = 10 Hex (Ch 5 = 1) When #AO2.9 Chs 1-8 are set to 0, the RTU will use 10 Hex (Ch 5 = 1) for the Low Voltage Wakeup Conditions.	Read/Write
AO2.9 Chs 9-16	Low Voltage Monitoring Interval 0 = Do not monitor supply voltage 1 to 255 = Monitor supply voltage every 1 to 255 minutes respectively	Default: 0 = low voltage shutdown disabled Set #AO2.9 = 256 or 16#100 (100 Hex) or similar in ladder logic to enable low voltage shutdown and monitoring of the supply every minute. Note: **5V Out** must be enabled to allow the LP-2/3 to monitor the supply voltage (please see #DO2.12)	Read/Write
AO2.10	Input Supply Shutdown/Startup Voltages Chs 1-8 (1-255 x 0.1 V) = LP-2/3 shutdown voltage Chs 9-16 (1-255 x 0.1 V) = LP-2/3 startup voltage	Default: AO2.10=7668 Hex Chs 1-8 = 104 (68H) = 10.4V shutdown Chs 9-16 = 118 (76H) = 11.8V startup	Read/Write

* The LP-2/3 can only measure temperatures 0 °C and above. The negative temperature limit allows for future development.

Hardware Setup Registers

Register	Description	Default	Comments	Read/Write
DI1.10	Port 3 or Port 4 detected	-	1 = one or two option ports detected	Read
DO1.13	Pulse input 1 enable	1	1 = enabled. Also allows wakeup via DI1	Read/Write
DO1.14	Pulse input 2 enable	1	1 = enabled. Also allows wakeup via DI2	Read/Write
DO1.15	Pulse input 1 active transition	0	0 = low to high transition, 1 = high to low	Read/Write
DO1.16	Pulse input 2 active transition	0	0 = low to high transition, 1 = high to low	Read/Write
DO2.7	DO3 enable	0	0 = DO3 disabled & DI5 enabled	Read/Write
DO2.8	DO4 enable	0	0 = DO4 disabled & DI6 enabled	Read/Write
DO2.9	DO5 enable	0	0 = DO5 disabled & DI7 enabled	Read/Write
DO2.10	DO6 enable	0	0 = DO6 disabled & DI8 enabled	Read/Write
DO2.11	**V1 Out** enable	1	1 = **V1 Out** enabled	Read/Write
DO2.12	I/O & **5V Out** disable	0	0 = I/O and **5V Out** enabled When disabled (1 or ON): • Switches off **5V Out** • Stops updating analog inputs and digital inputs 3 to 8 • Analog output is set to 0 (LP-3 only) • Digital inputs 1 & 2 and Digital outputs 1& 2 still operate • Disables power to ports 3 and 4 Note 1: I/O & **5V Out** must be enabled for low voltage shutdown to operate correctly (allows reading of supply voltage) Note 2: **CPU 5V Out** remains enabled	Read/Write
DO2.13	**24V Out** enable	1	1 =**24V Out** enabled	Read/Write
DO2.14	Port 3 and Port 4 disable	0	0 = Port 3 and Port 4 enabled	Read/Write
AO5.5	Pulse Output count	0	Uses DO Ch3 (must be enabled)	Read/Write
AO5.6	Pulse ON time	0	x 1ms	Read/Write
AO5.7	Pulse OFF time	0	x 1ms	Read/Write
DO6.1	Analog input 1 live zero	0	0=0-5 V or 0-20 mA, 1=1-5 V or 4-20 mA*	Read/Write
DO6.2	Analog input 2 live zero	0	0=0-5 V or 0-20 mA, 1=1-5 V or 4-20 mA*	Read/Write
DO6.3	Analog input 3 live zero	0	0=0-5 V or 0-20 mA, 1=1-5 V or 4-20 mA*	Read/Write
DO6.4	Analog input 4 live zero	0	0=0-5 V or 0-20 mA, 1=1-5 V or 4-20 mA*	Read/Write
AO3.9, AO3.10	Pulse 1 (DI1) or pulse 2 (DI2) debounce time respectively	0	0 to 32000 ms debounce time The pulse must remain in the active state for the debounce time before a pulse is counted (0 = no debounce)	Read/Write
AO4.2, AO4.3, AO4.4, AO4.5	Analog input 1 to 4 average sample count respectively	1	1-59 samples. Number of samples to average the analog input over before updating the analog input data register. Each sample takes up to 110 ms.	Read/Write

* Analog inputs can be changed from voltage to current inputs by installing an external 250 Ω resistor.

Miscellaneous Data

Register	Description	Comments	Read/Write
AI2.7	Port 3 option board type detected	0=none, 1=serial/fibre optic/spread spectrum radio/new pstn, 2=old pstn, 3=pline, 4=ethernet, 5=image, 6=image (running)	Read
AI2.8	Port 4 option board type detected	0=none, 1=serial/fibre optic/spread spectrum radio/new pstn, 2=old pstn, 3=pline, 4=ethernet, 5=image, 6=image (running)	Read
AO2.13	Ports 1 and 2 Data Format	Ch 1: Port 1 stop bits, 0=one, 1=two Ch 2: Port 1 parity, 0=even, 1=odd Ch 3: Port 1 parity enable, 0=disable, 1=enable Ch 4: Port 1 data bits, 0=eight, 1=seven Ch 5-8: Port 1 new data format enable. Must write D hex (1101 binary) to enable. Ch 9: Port 2 stop bits, 0=one, 1=two Ch 10: Port 2 parity, 0=even, 1=odd Ch 11: Port 2 parity enable, 0=disable, 1=enable Ch 12: Port 2 data bits, 0=eight, 1=seven Ch 13-16: Port 2 new data format enable. Must write D hex (1101 binary) to enable.	Read/Write
AO2.14	Ports 3 and 4 Data Format	Ch 1: Port 3 stop bits, 0=one, 1=two Ch 2: Port 3 parity, 0=even, 1=odd Ch 3: Port 3 parity enable, 0=disable, 1=enable Ch 4: Port 3 data bits, 0=eight, 1=seven Ch 5-8: Port 3 new data format enable. Must write D hex (1101 binary) to enable. Ch 9: Port 4 stop bits, 0=one, 1=two Ch 10: Port 4 parity, 0=even, 1=odd Ch 11: Port 4 parity enable, 0=disable, 1=enable Ch 12: Port 4 data bits, 0=eight, 1=seven Ch 13-16: Port 4 new data format enable. Must write D hex (1101 binary) to enable.	Read/Write
AI3.5	Vref	ADC reference voltage. 0-32767 = 0-5V. Normally 2.5V ±2%	Read
DI3.1	Firmware error	1 = Error	Read
DI3.2	SRAM error	1 = Error	Read

7. Specifications

An LP-2 has no analog output and one option port. An LP-3 has one analog output and two option ports.

Power and Environment

Supply (**+VDC IN**)	9.0 to 15.0 VDC @ 2 A maximum	
Backup Battery (**+BAT**)	Supported. LP-2/3 provides current-limited trickle charging. Protection: 1.85 A self-resetting polyfuse.	
Low Voltage Shutdown	Supported but disabled by default. To enable, set #AO2.9=16#100 (100 Hex) using ladder logic. The RTU will then shutdown at 10.4 V ± 2% and startup at 11.8 V ± 2% by default. Note: Shutdown and Startup voltage settings are configurable.	
Supply Fuse	1.85 A (self-resetting polyfuse)	
Power Consumption	Sleep mode: < 2 mA (can also count DI1 & DI2 pulses in this mode) Running, no option ports, output voltages and IO enabled: < 55 mA	
Operating Temperature Range	-20 to +70 °C	
Storage Temperature range	-40 to +85 °C	
Operating Humidity	5 to 98 % R.H. Non Condensing	
Output Voltages	**5V Out**	5V @ 10 mA. Software controllable. Current limited by 100 Ω series resistor.
	CPU 5V Out	5V @ 10 mA. Always enabled.
	V1 Out	+VDC IN (supply voltage) @ 100 mA continuous and 1 A intermittent (1% duty cycle). Software controllable. 1.35 A self-resetting polyfuse.
	24V Out	+24 V @ 100 mA. Software controllable. Self-resetting polyfuse.
Clock and SRAM Battery Backup	5 years (RTU unpowered)	
Monitoring	Supply voltage and RTU temperature	

Processor

CPU	Hitachi H8S/2144 operating at 32 kHz or 7.38 MHz
Word Size	8 Bit external data bus / 16 Bit internal data Bus.
RAM – Flash	128 kB for firmware and 4096 kB (4 MB expanded memory) for event logs. Notes: • Event logs stored in expanded memory can only be uploaded using Toolbox 32. • Firmware drivers are stored in SRAM.
RAM – Static CMOS (SRAM)	512 kB (battery backed). Removing the battery link for 5 minutes will clear the configuration and firmware drivers. Note: the clock needs to be set after the battery link is removed. (128 kB is reserved for firmware use, 384 kB user configurable)
Event Logs	32,000 standard - can be uploaded using the Kingfisher driver in Citect, Wonderware and ClearSCADA 300,000 - when using expanded memory. These event logs can be uploaded using Toolbox 32 only
Real-time Clock	Active while sleeping and when RTU is powered off. Accuracy: ± 1 minute / month
Watchdog timer	Triggered by hardware watchdog 5 second time-out or supply voltage below ~4.5 V
Battery Type	Lithium Primary (Not rechargeable).
Battery Life - module unpowered	7 years @ 25 °C
Battery Replacement	At above intervals
LED Status Indication	Flashing ‘Health’ LED: Flashes every ~1 seconds when awake and running normally Flashes every ~4 seconds when asleep Flashes every ~0.4 seconds when ladder error (eg continuous loop)
Self testing	After a cold start the SRAM, firmware CRC and clock are checked. The firmware is also checked after a download.

Communications Ports

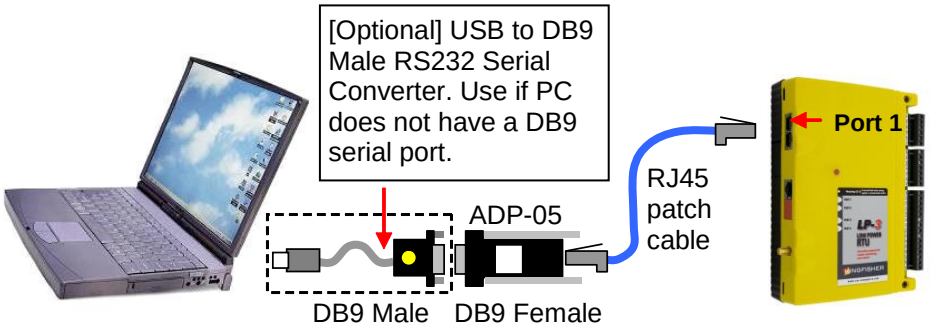
Port 1	RS232C 300 to 38400 bps Non-isolated	
	Data format	8 data bits, no parity, 1 stop (8, N, 1)
	Isolation	None
Port 2	RS232C/RS422/RS485 300 to 38400 bps Isolated	
	Output Power	+4.3 V @ 100 mA
	RS422 Line drive	10 Receivers
	RS485 Line drive	32 Receivers
	Data format	8 data bits, no parity, 1 stop (8, N, 1)
	Isolation	2.5 kV
Port 3 Port 4	Option boards supported: Isolated Serial (RS232/RS422/RS485 300 to 115200 bps), PSTN (33.6 kbps), Line-2 2/4 Wire Leased Line or Analog Radio*, Fibre Optic and Spread Spectrum Radio. Ethernet, HART and Image Capture option boards are NOT supported. Full specifications are detailed in the Kingfisher PLUS+ Hardware manual (available from http://helpdesk.servelec-semaphore.com). *Note: When an external radio is used with the Line-2 option board, the carrier detect (CD) input on the RJ45 is not used. Carrier detect is based on detection of tones.	

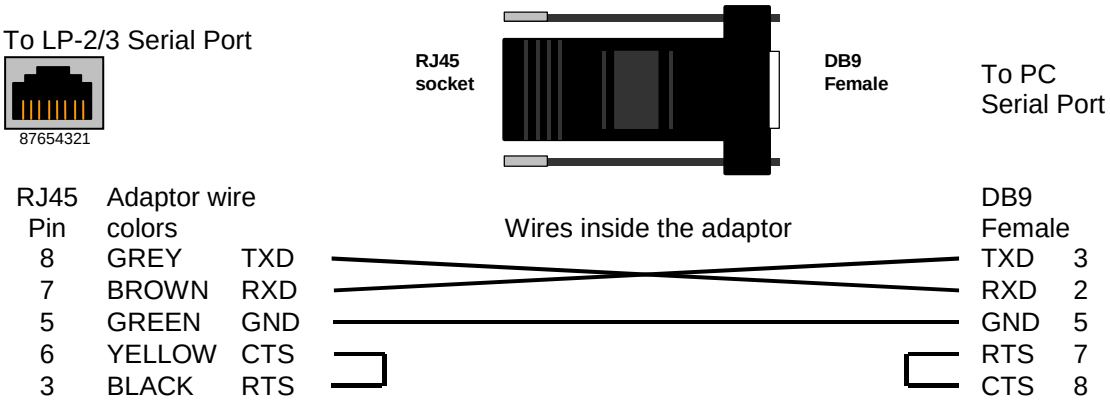
RJ45 Pin	Port 1 or Port 2 RS232C	Port 2 RS422 / RS485 *
1	DTR	N/A
2	4.3 V	N/A
3	RTS	TX+
4	DCD	N/A
5	GND (0 V)	GND
6	CTS	RX+
7	RXD	RX-
8	TXD	TX-

* Wire TX- and RX- together to obtain LINE- (RS485). Wire TX+ and RX+ together to obtain LINE+ (RS485).

PC to LP-2/3 Configuration Cable (Creates a null modem cable)

The ADP-05 adaptor can be ordered from your Kingfisher supplier.

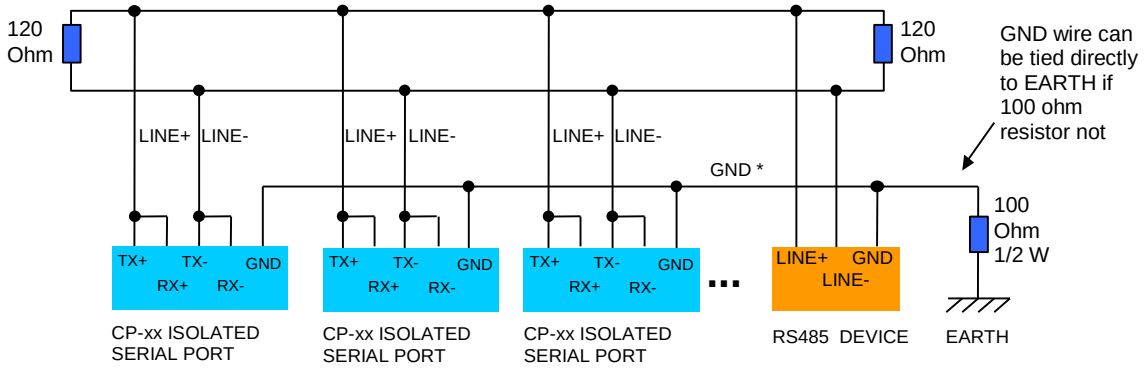

[Optional] USB to DB9 Male RS232 Serial Converter. Use if PC does not have a DB9 serial port.
ADP-05
RJ45 patch cable
Port 1
DB9 Male
DB9 Female



RS485 Wiring Diagram

(Can use Port 2 or an Isolated Serial Option Board for RS485 communications)

Each RTU can transmit/receive to any RTU, one at a time. Up to 32 RTUs (or other RS485 devices) can be connected to the RS485 bus. Maximum RS485 Wire Length: 600 m.

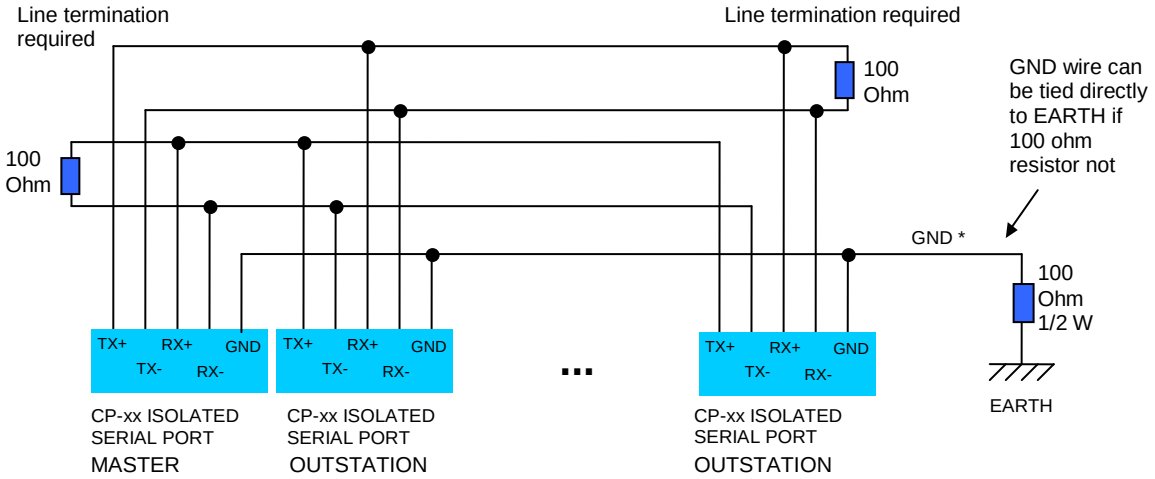


* If GND wire not available, each GND terminal can be tied to EARTH locally using a 100 ohm 1/2 W resistor or directly to EARTH (if resistors not available).

RS422 4-Wire Wiring Diagram

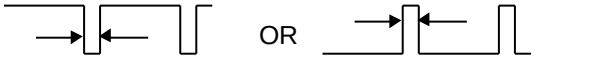
(Can use Port 2 or an Isolated Serial Option Board for RS422 communications)

The master RTU can transmit/receive to any one outstation RTU at any time or each outstation can transmit/receive to the master RTU, one at a time. Note: outstation RTUs cannot communicate with each other. Up to 10 RTUs (or other RS422 devices) can be connected to the RS422 bus. Maximum RS422 Wire Length: 1200 m



* If GND wire not available, each GND terminal can be tied to EARTH locally using a 100 ohm 1/2 W resistor or directly to EARTH (if resistors not available).

Digital Inputs

Digital Inputs	8 Max. Channels 5-8 can be configured as digital outputs.
Wakeup Functionality	Channels 1 & 2 can wake the RTU on change of state if enabled
Pulse Counting (digital inputs 1 and 2)	100 Hz max. in subactive powerdown mode 1 kHz max. in run mode 16-bit counter (can count up to 65535 pulses) Configurable positive or negative edge counting Pulse must be asserted for at least 0.5 ms in run mode or for 5 ms in subactive powerdown mode. **Negative Edge Counting** OR **Positive Edge Counting**  0.5 ms min. 0.5 ms min.
Input Voltage Range	5.0 to 30 VDC = ON 0 to 1.0 VDC = OFF
Current Consumption	<1 mA for all 8 inputs ON @ 5 VDC
Isolation	None
Debounce	0 to 32,000 ms (configurable). Software debounced by reading 3 times and then accepting the change if all readings are the same

Digital Outputs

	Relay Outputs	**Transistor Outputs**
Digital Outputs	2	0 to 4 Up to 4 digital inputs can be configured as sinking transistor digital outputs
Relay Type	SPST-NO (normally open)	
Pulse output		Uses DO channel 3 0 to 65535 pulses 500 Hz maximum Configurable pulse ON and OFF times (0 to 65535 ms)
Maximum Switching voltage	30 VAC, 30 VDC	30 VDC
Maximum Switching Current	2 A	300 mA total for all 4 outputs
Maximum Switching Power	60 VA, 60 W	
Isolation	Transient voltage: 500 V Maximum working voltage in respect to system earth/ground must not exceed SELV limits (42.4 Vpeak / 60 VDC)	No isolation
Operating Power	300 mW at 12 V	
Fuse		300 mA fuse common for the 4 outputs
Effect of RTU power down	Hold last state	Last state restored on power up

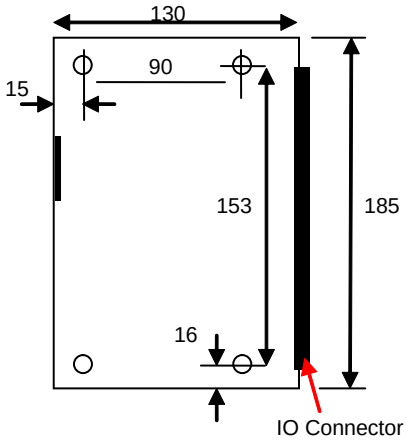
Analog Inputs

Analog Inputs	4
Input Voltage Range	0-5 V (default) or 1-5 V (software configurable)
Input Current Range	0-20 mA (default) or 4-20 mA (software configurable) Note: external 250 ohm resistor must be installed for current inputs. Please see wiring diagram for details.
A/D Converter Resolution	12 bit
Binary input range	0 to 32760
Update Rate	5 ms
Accuracy	± 0.2% @ 25 °C (Voltage Input) ± (0.2 + x) % @ 25 °C (Current Input) x=% accuracy of the external resistor. Eg. Using 0.1% accuracy resistors (as supplied with the LP-2/3): Accuracy = ± 0.3% @ 25 °C
Isolation	None
Input Impedance	10 kΩ
Debounce	Each analog channel is averaged over a configurable number of samples. The current analog value is updated each I/O scan

Analog Output (LP-3 only)

Analog Outputs	1
Output Ranges	0-5 V or 1-5 V from **AOV** output (software configurable) 0-20 mA or 4-20 mA from **AOC** output (software configurable)
D/A Converter Resolution	15 bit
Binary Input Range	0 to 32767
Update Rate	5 ms
Accuracy (voltage and current)	± 0.5% @ 25 °C
User Load	850 Ω maximum for current output 1 kΩ minimum for voltage output
Isolation	500 V RMS channel to logic
Output Protection	Protected against continuous short circuit.

Mounting Details



The LP-2/3 RTU has a two part enclosure comprised of a plastic base and a cover. The overall dimensions of the LP-2/3 RTU are 185 mm H x 130 mm W x 50 mm D. The enclosure includes room behind the circuit board to allow the housing of third party devices. Available space is approximately 160 mm H x 120 mm W x 20 mm D.

The LP-2/3 enclosure is mounted by fastening 4 screws (up to 3.5 mm in diameter) through the base plate. Once the base plate is mounted, the cover of the LP-2/3 is then attached.

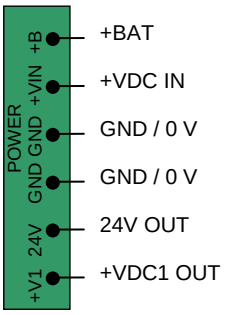
Dimensions

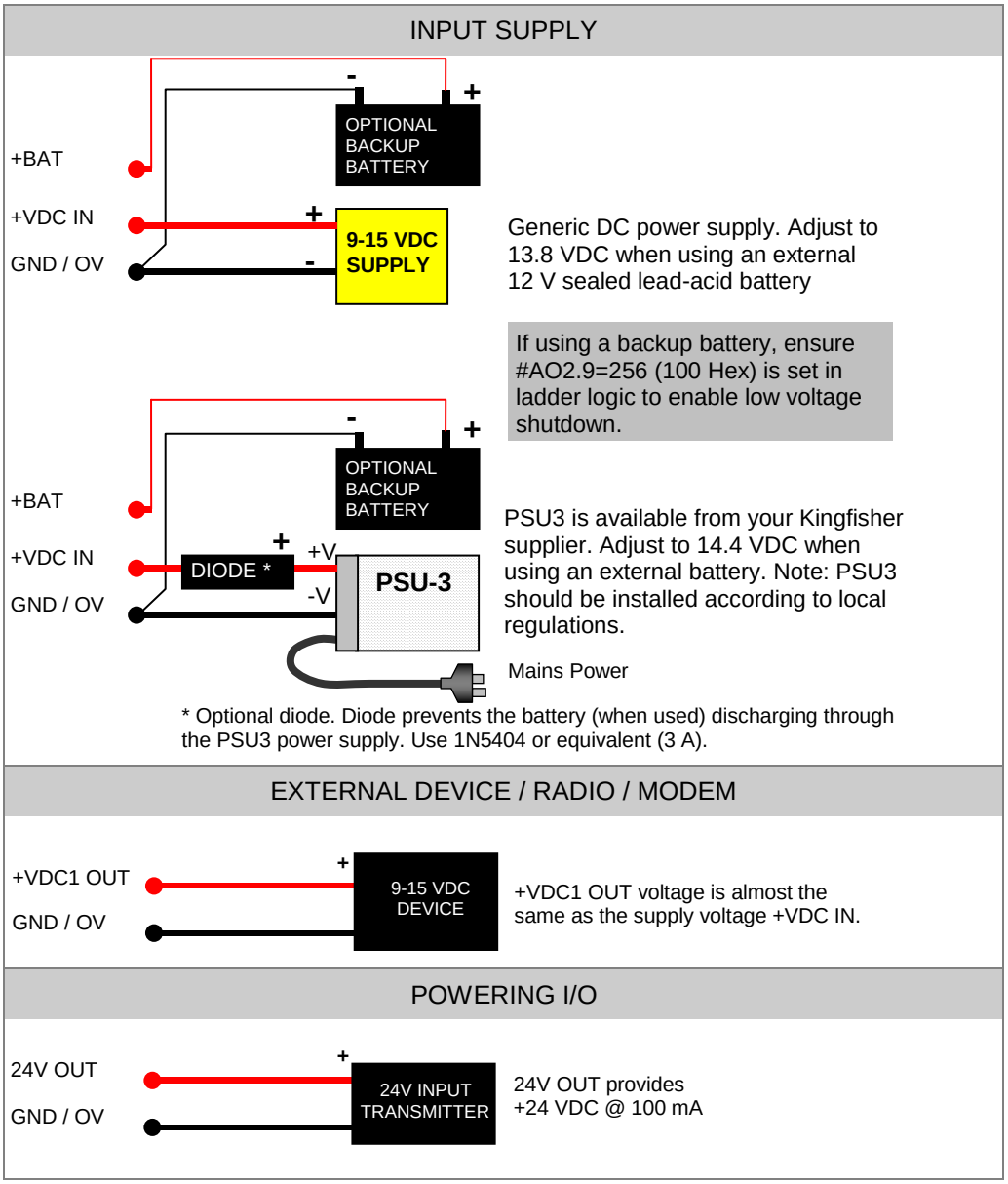
Enclosure overall size	185 mm H x 130 mm W x 50 mm D
Enclosure cavity (for OEM device)	160 mm H x 120 mm W x 20 mm D

8. Wiring Details

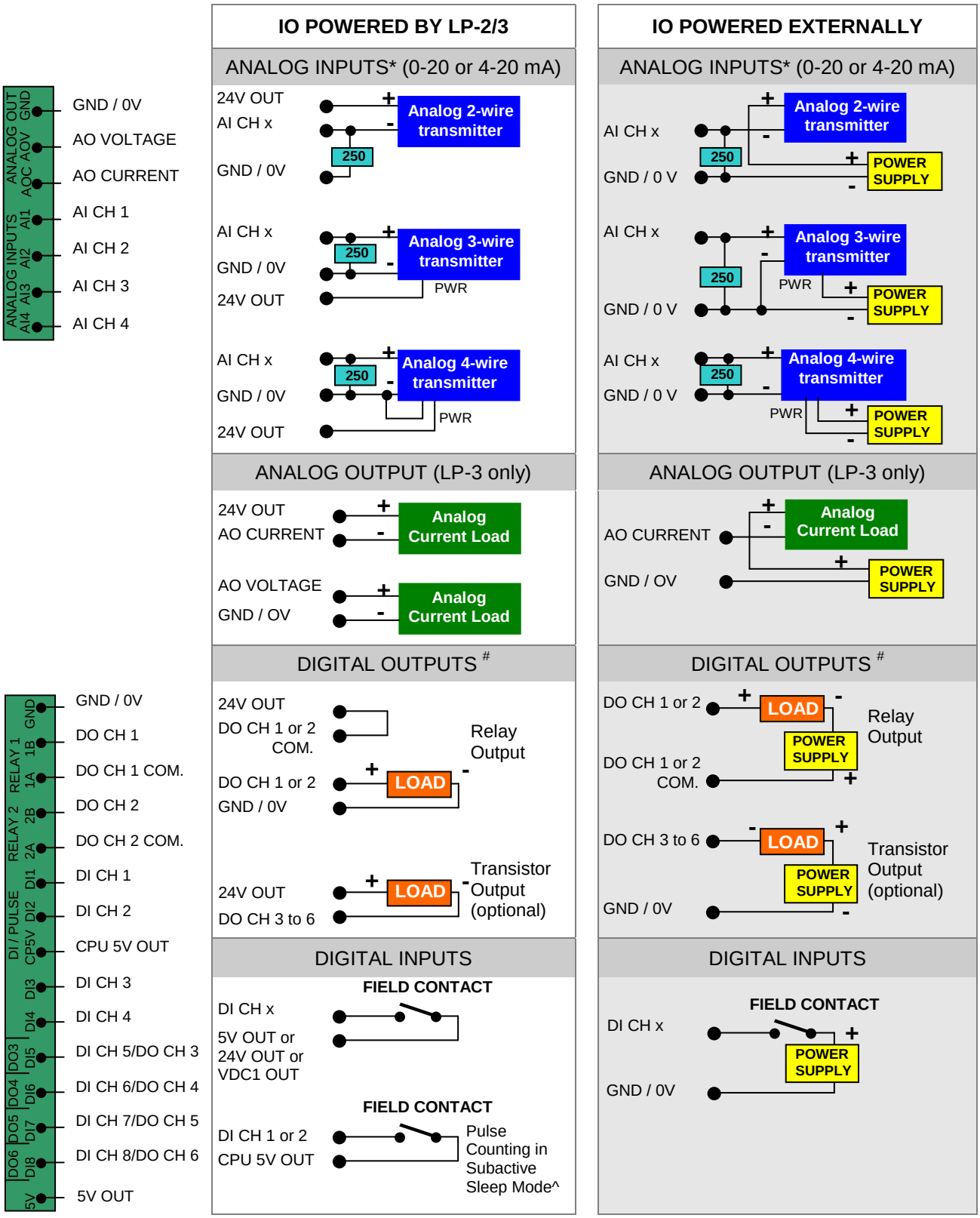
The LP-2/3 has three connectors used for wiring power, analog and digital inputs and outputs. Each connector is a different size and is identified by the labeling on the connector.

Power Wiring




INPUT SUPPLY
+BAT
+VDC IN
GND / OV
OPTIONAL BACKUP BATTERY
9-15 VDC SUPPLY
Generic DC power supply. Adjust to 13.8 VDC when using an external 12 V sealed lead-acid battery
If using a backup battery, ensure #AO2.9=256 (100 Hex) is set in ladder logic to enable low voltage shutdown.
+BAT
+VDC IN
GND / OV
OPTIONAL BACKUP BATTERY
DIODE *
+V
-V
PSU-3
PSU3 is available from your Kingfisher supplier. Adjust to 14.4 VDC when using an external battery. Note: PSU3 should be installed according to local regulations.
Mains Power
* Optional diode. Diode prevents the battery (when used) discharging through the PSU3 power supply. Use 1N5404 or equivalent (3 A).
EXTERNAL DEVICE / RADIO / MODEM
+VDC1 OUT
GND / OV
9-15 VDC DEVICE
+VDC1 OUT voltage is almost the same as the supply voltage +VDC IN.
POWERING I/O
24V OUT
GND / OV
24V INPUT TRANSMITTER
24V OUT provides +24 VDC @ 100 mA

IO Wiring



* When using a 0-5 V or 1-5 V analog input, the external 250 Ω resistor is not required. Unused inputs should be tied to GND / 0V to prevent spurious readings. 0-5 or 0-20mA is the default analog input range. 1-5 or 4-20mA is software configurable.

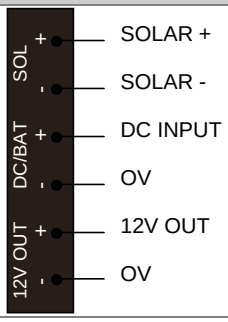
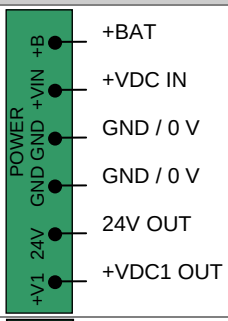
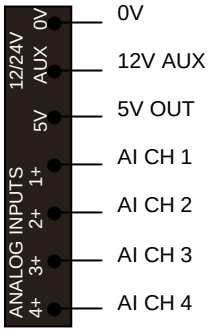
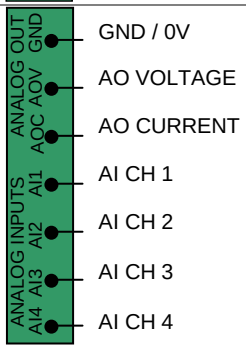
DI Channels 5 to 8 can be software configured as transistor digital outputs. When the output is energised, the output is shorted to the OV terminal.

^ **CPU 5V Out** stays enabled during power down modes and can only be used with DI1 and DI2 to wake the RTU or to count pulses while remaining asleep. Note: **CPU 5V Out** is not isolated.

9. Appendices

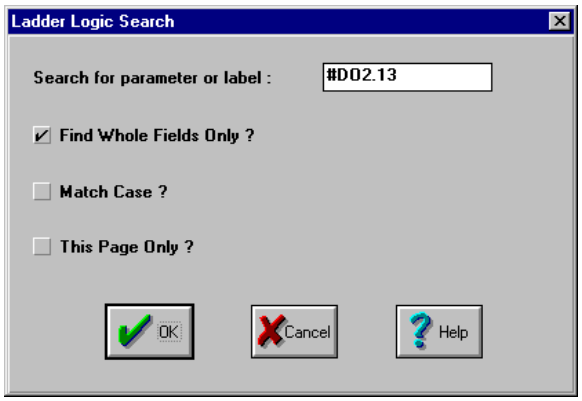
Appendix - Replacing an LP-1 with an LP-2/3

Please consider the following differences:

Item	LP-1	LP-2/3
Power Wiring		
Analog Wiring		
#DI1.9	0 [default] = port 3 disabled	Always 0 (no longer used)
#DI1.10	0 [default] = port 4 disabled	1 = port 3 or port 4 detected
#DO2.11	0 [default] = disable **12V Out**	1 [default] = enable **V1 Out**
#DO2.13	1 [default] = disable port 3	1 [default] = enable **24V Out**
#DO2.14	1 [default] = disable port 4	0 [default] = enable ports 3 and 4
#AI1.8	Supply Voltage 0 to 32760 = 0 to *25* V	Supply Voltage 0 to 32760 = 0 to *20.15* V
#AI2.7	Trio SR radio TX power	Port 3 option board type
#AO2.9	0 [default] = shutdown at 10.4 V and check supply every 5 minutes	38 Hex [default] = disable low voltage shutdown

NOTES:

- LP-1 configurations that use port 3 (Trio analog radio or internal GSM) typically clear #DO2.13 (port 3 disable) when ladder logic is first scanned. Clearing #DO2.13 in an LP-2/3 will disable **24V Out**.
- If using a backup battery, ensure #AO2.9 is set to 256 (100 Hex) in ladder logic to enable low voltage shutdown and checking of the supply every minute.



To check if any of the above registers are used in ladder logic, view the logic using Toolbox 32. Then select **Ladder, Find** (or press CTRL+F). Enter each of the above registers in the **Search for parameter or label** field as shown.